Z8 Family of Microcontrollers

Z8® CPU

User Manual

UM001604-0108

⚠ **Warning:** DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Revision History

Each instance in Revision History reflects a change to this document from its previous revision. For more details, refer to the corresponding pages and appropriate links in the table below.

Date	Revision Level	Description	Page No
January 2008	04	Updated Zilog logo, Zilog text, Disclaimer section, and implemented Style Guide.	All
February 2007	03	Changed the OP code to B0 and B1 in Instruction Description.	167
September 2004	02	Formatted to current publication standards.	All pages

Table of Contents

Z8® CPU Product Overview

Zilog's Z8® microcontroller (MCU) product line continues to expand with new product introductions. Zilog MCU products are targeted for cost-sensitive, high-volume applications including consumer, automotive, security, and HVAC. It includes ROM-based products geared for high-volume production (where software is stable) and one-time programmable (OTP) equivalents for prototyping as well as volume production where time to market or code flexibility is critical (see Table 1 on page 3). A variety of packaging options are available including plastic DIP, SOIC, PLCC, and QFP.

A generalized Z8 CPU block diagram is displayed in Figure 1 on page 2. The same on-chip peripherals are used across the MCU product line with the primary differences being the amount of ROM/RAM, number of I/O lines present, and packaging/temperature ranges available. This allows code written for one MCU device to be easily ported to another family member.

Key Features

The key features include:

General-Purpose Register File—Every RAM register acts like an accumulator, speeding instruction execution and maximizing coding efficiency. Working register groups allow fast context switching.

Flexible I/O—I/O byte, nibble, and/or bit programmable as inputs or outputs. Outputs are software programmable as open-drain or push–pull on a port basis. Inputs are Schmitt-Triggered with autolatches to hold unused inputs at a known voltage state.

Analog Inputs—Three input pins are software programmable as digital or analog inputs. When in analog mode, two comparator inputs are provided with a common reference input. These inputs are ideal for a variety of common functions, including threshold level detection, analog-to-digital conversion, and short circuit detection. Each analog input provides a unique maskable interrupt input.

Timer/Counter—The Timer/Counter (T/C) consists of a programmable 6-bit prescaler and 8-bit downcounter, with maskable interrupt upon end-of-count. Software controls T/C load/start/stop, countdown read (at any time on the fly), and maskable end-of-count interrupt. Special functions available include T_{IN} (external counter input, external gate input, or external trigger input) and T_{OUT} (external access to timer output or the internal system clock). These special functions allow accurate hardware input pulse measurement and output waveform generation.

Interrupts—There are six vectored interrupt sources with software-programmable enable and priority for each of the six sources.

Watchdog Timer—An internal Watchdog Timer (WDT) circuit is included as a fail-safe mechanism so that if software strays outside the bounds of normal operation, the WDT is used to time-out and reset the MCU. To maximize circuit robustness and reliability, the

default WDT clock source is an internal RC circuit (isolated from the device clock source).

Auto Reset/Low-Voltage Protection—All family devices have internal Power-On Reset. ROM devices add low-voltage protection. Low-voltage protection ensures the MCU is in a known state at all times (in active RUN or RESET modes) without external hardware (or a device reset pin).

Low-EMI Operation—Mode is programmable via software or as a mask option. This new option provides for reduced radiated emission via clock and output drive circuit changes.

Low-Power—CMOS with two standby modes; STOP and HALT.

Full Z8® Instruction Set—Forty-eight basic instructions, supported by six addressing modes with the ability to operate on bits, nibbles, bytes, and words.
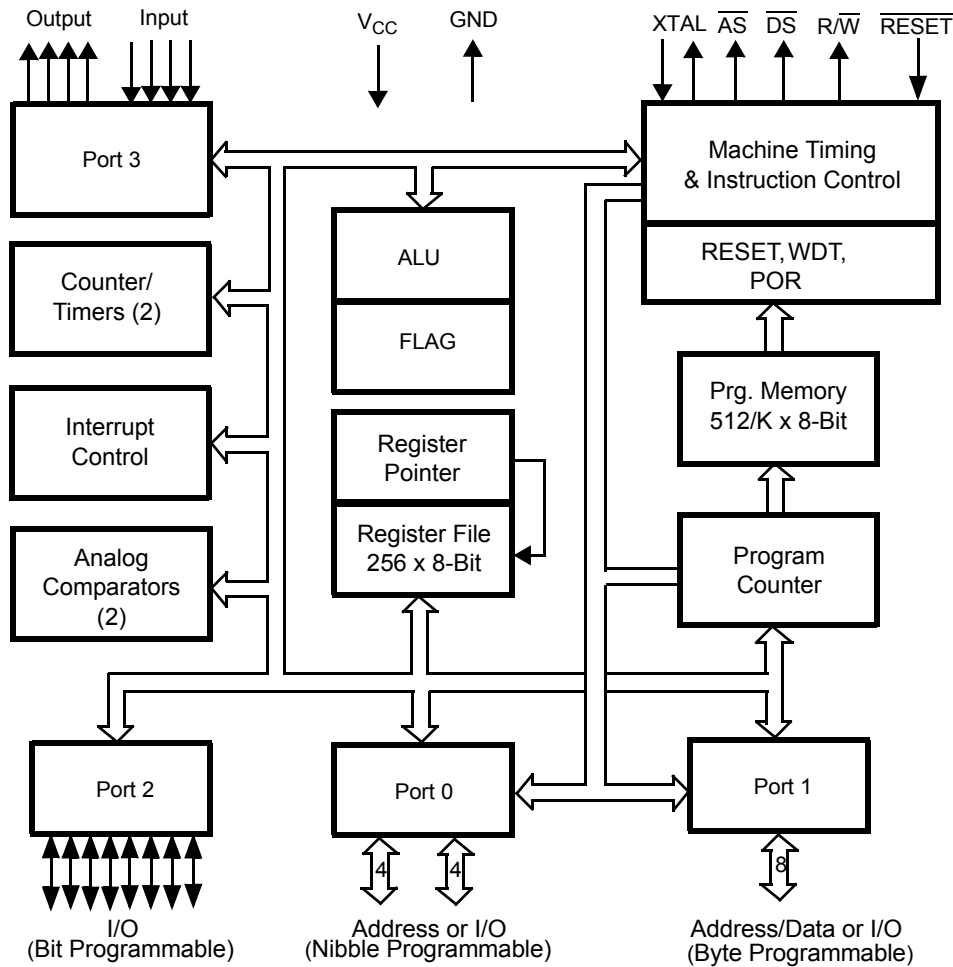
Figure 1. Z8 CPU Block Diagram

Product Development Support

The Z8® MCU product line is fully supported with a range of cross assemblers, C compilers, ICEBOX emulators, single and gang OTP/EPROM programmers, and software simulators.

The Z86CCP01ZEM low-cost Z8 CCP real-time emulator/programmer kit is designed specifically to support all the products outlined in Table 1 on page 3.

Table 1. Zilog General-Purpose Microcontroller Product Family

Product	ROM/ RAM	I/O	T/C	AN	INT	WDT	POR	V_{BO}	RC	Speed (MHz)	Pin Count
Z86C03	512/60	14	1	2	6	F	Y	Y	Y	8	18
Z86E03	512/60	14	1	2	6	F	Y	N	Y	8	18
Z86C04	1K/124	14	2	2	6	F	Y	Y	Y	8	18
Z86E04	1K/124	14	2	2	6	F	Y	N	Y	8	18
Z86C06	1K/124	14	2	2	6	P	Y	Y	Y	12	18
Z86E06	1K/124	14	2	2	6	P	Y	N	Y	12	18
Z86C08	2K/124	14	2	2	6	F	Y	Y	Y	12	18
Z86E08	2K/124	14	2	2	6	F	Y	N	Y	12	18
Z86C30	4K/236	24	2	2	6	P	Y	Y	Y	12	28
Z86E30	4K/236	24	2	2	6	P	Y	N	Y	12	28
Z86C31	2K/124	24	2	2	6	P	Y	Y	Y	8	28
Z86E31	2K/124	24	2	2	6	P	Y	N	Y	8	28
Z86C40	4K/236	32	2	2	6	P	Y	Y	Y	16	40/44
Z86E40	4K/236	32	2	2	6	P	Y	N	Y	16	40/44

Note: Z86Cxx signify ROM devices; 86xx signify EPROM devices; F = fixed; P = programmable.

The Z86CCP01ZEM kit includes:

- Z8 CCP evaluation board

- Z8 CCP power cable

- Zilog Developer Studio (ZDS) CD-ROM, Including Windows-Based GUI Host Software

- 1999 Zilog Technical Library

- Z8 CCP User Manual

A Z8 CCP Emulator Accessory Kit (Z8CCP00ZAC) is also available and provides an RS-232 cable and power cable along with the 28- and 40- pin ZIF sockets and 28- and 40-pin target connector cables required to emulate/program 28-/40-pin devices.

Address Space

Introduction

Z8® CPU includes the following four address spaces:

- The Z8 Standard Register File contains addresses for peripheral, control, all general-purpose, and all I/O port registers. This is the default register file specification.

- The Z8 Expanded Register File (ERF) contains addresses for control and data registers for additional peripherals/features.

- Z8 external Program Memory contains addresses for all memory locations having executable code and/or data.

- Z8 external data memory contains addresses for all memory locations that hold data only, whether internal or external.

Z8® CPU Standard Register File

The Z8 Standard Register File totals up to 256 consecutive bytes (Registers). The register file consists of 4 I/O ports (`00h-03h`), 236 General-Purpose Registers (`04h-EFh`), and 16 control registers (`F0h-FFh`). Table 2 lists the layout of the register file, including register names, locations, and identifiers.

Table 2. Z8 Standard Register File

Hex Address	Register Identifier	Register Description
FF	SPL	Stack Pointer Low Byte
FE	SPH	Stack Pointer High Byte
FD	RP	Register Pointer
FC	FLAGS	Program Control Flags
FB	IMR	Interrupt Mask Register
FA	IRQ	Interrupt Request Register
F9	IPR	Interrupt Priority Register
F8	P01M	Port 0–1 Mode Register
F7	P3M	Port 3 Mode Register
F6	P2M	Port 2 Mode Register
F5	PRE0	T0 Prescaler

Table 2. Z8 Standard Register File (Continued)

Hex Address	Register Identifier	Register Description
F4	T0	Timer/Counter 0
F3	PRE1	T1 Prescaler
F2	T1	Timer/Counter 1
F1	TMR	Timer Mode
F0	SIO	Serial I/O
EF	R239	
General-Purpose Registers (GPR)		
04	R4	
03	P3	Port 3
02	P2	Port 2
01	P1	Port 1
00	P0	Port 0

Registers can be accessed as either 8-bit or 16-bit registers using Direct, Indirect, or Indexed Addressing. All 236 general-purpose registers can be referenced or modified by any instruction that accesses an 8-bit register, without the requirement for special instructions. Registers accessed as 16 bits are treated as even-odd register pairs (there are 118 valid pairs). In this case, the data's most significant byte (MSB) is stored in the even numbered register, while the least significant byte (LSB) goes into the next higher odd numbered register. See Figure 2.

MSB	LSB
Rn	Rn+1

n = Even Address
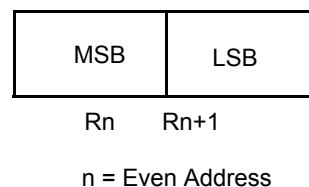
Figure 2. 16-Bit Register Addressing

By using a logical instruction and a mask, individual bits within registers can be accessed for bit set, bit clear, bit complement, or bit test operations. For example, the instruction AND R15, MASK performs a bit clear operation, Figure 3 on page 7 displays this example.

| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | R15

| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | MASK

AND R15, DFh   ;Clear Bit 5 of Working Register 15

| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | R15
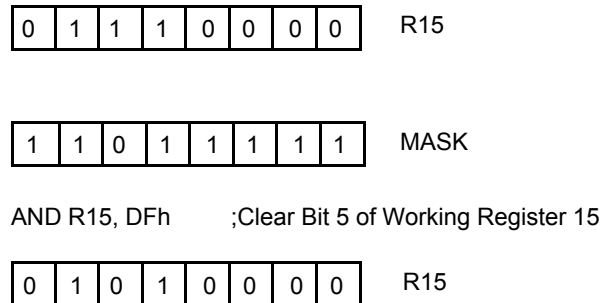
Figure 3. Accessing Individual Bits (Example)

When instructions are executed, registers are read when defined as sources and written when defined as destinations. All General-Purpose Registers function as accumulators, address pointers, index registers, stack areas, or scratch pad memory.

General-Purpose Registers

General-Purpose Registers are undefined after the device is powered up. The registers keep their last value after any reset, as long as the reset occurs in the V_{CC} voltage-specified operating range. It does not keep its last state from a V_{LV} reset if V_{CC} drops below 1.8 V.

▶ **Note:** *Registers in Bank* E0-EF *may only be accessed through the working register and indirect addressing modes. Direct access cannot be used because the 4-bit working register address mode already uses the format* [E | dst]*, where* dst *represents the working register number from* 0h *to* Fh*.*

RAM Protect

The upper portion of the register file address space 80h to EFh (excluding the control registers) may be protected from reading and writing. The RAM Protect bit option is mask-programmable and is selected by the customer when the ROM code is submitted. After the mask option is selected, activate this feature from the internal ROM code to turn OFF/on the RAM Protect by loading either a 0 or 1 into the IMR register, bit D6. A 1 in D6 enables RAM Protect. Only devices that use registers 80h to EFh offer this feature.

Working Register Groups

Z8® instructions can access 8-bit registers and register pairs (16-bit words) using either 4-bit or 8-bit address fields. 8-bit address fields refer to the actual address of the register. For example, Register 58h is accessed by calling upon its 8-bit binary equivalent, 01011000 (58h).

With 4-bit addressing, the register file is logically divided into 16 Working Register Groups of 16 registers each, as listed in Table 3. These 16 registers are known as Working Registers. A Register Pointer (one of the control registers, FDh) contains the base address of the active Working Register Group. The high nibble of the Register Pointer determines the current Working Register Group.

When accessing one of the Working Registers, the 4-bit address of the Working Register is combined within the upper four bits (high nibble) of the Register Pointer, thus forming the 8-bit actual address. Figure 4 on page 9 displays this operation. Because working registers are typically specified by short format instructions, there are fewer bytes of code required, which reduces execution time. In addition, when processing interrupts or changing tasks, the Register Pointer speeds context switching. A special Set Register Pointer (SRP) instruction sets the contents of the Register Pointer.

Table 3. Working Register Groups

Register Pointer (FDh) High Nibble	Working Register Group (Hex)	Actual Registers (Hex)
1111b	F	F0–FF
1110b	E	E0–EF
1101b	D	D0–DF
1100b	C	C0–CF
1011b	B	B0–BF
1010b	A	A0–AF
1001b	9	90–9F
1000b	8	80–8F
0111b	7	70–7F
0110b	6	60–6F
0101b	5	50–5F
0100b	4	40–4F
0011b	3	30–3F
0010b	2	20–2F
0001b	1	10–1F
0000b	0	00–0F

| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Register Pointer (FDh), Standard Register File

| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

INC R6 (instruction, short format)

| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

Actual register address (76h)
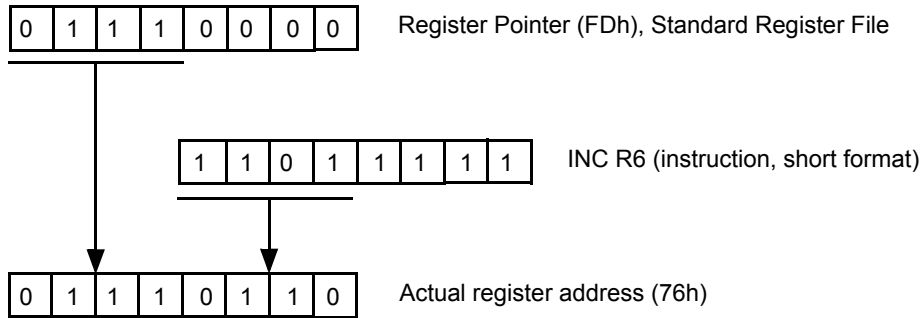
Figure 4. Working Register Addressing Examples

| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

R253
(Register Pointer)

The upper nibble of the register file address,
provided by the register pointer, specifies
the active working-register group.

FF
F0   Working Register Group F
EF
80
7F
70
6F
60
5F
50
4F
40
3F
30   Specified Working Register Group
2F
20
1F
10   Working Register Group 1
0F   Working Register Group 0
00   I/O Ports

The lower nibble
of the register
file address
(provided by the
instruction) points
to the specified
register.

R15 to R0

R15 to R4

R3 to R0

*Note: The full register file is shown. Refer to the selected device product specification for actual file
size.

Figure 5. Register Pointer

Error Conditions

Registers in the Z8® Standard Register File must be correctly used because certain conditions produce inconsistent results and should be avoided.

- Registers `F3h` and `F5h–F9h` are write-only registers. If an attempt is made to read these registers, `FFh` is returned. Reading any write-only register returns `FFh`.

- When register `FDh` (Register Pointer) is read, the least significant four bits (lower nibble) indicate the current Expanded Register File Bank. (For example, `0000` indicates the Standard Register File, while `1010` indicates Expanded Register File Bank A.)

- When Ports 0 and 1 are defined as address outputs, registers `00h` and `01h` return 1s in each address bit location when reading.

- Writing to bits that are defined as timer output, serial output, or handshake output has no effect.

- The Z8 CPU instruction `DJNZ` uses any general-purpose working register as a counter.

- Logical instructions such as OR and AND require that the current contents of the operand be read. They therefore do not function properly on write-only registers.

- The WDTMR register must be written within the first 60 internal system clocks (SCLK) of operation after a reset.

Z8 Expanded Register File

The standard register file of the Z8 CPU has been expanded to form 16 Expanded Register File (ERF) Banks, as displayed in Figure 6 on page 11. Each ERF Bank consists of up to 256 registers (the same amount as in the Standard Register File) that can then be divided into 16 Working Register Groups. This expansion allows for access to additional feature/peripheral control and data registers.
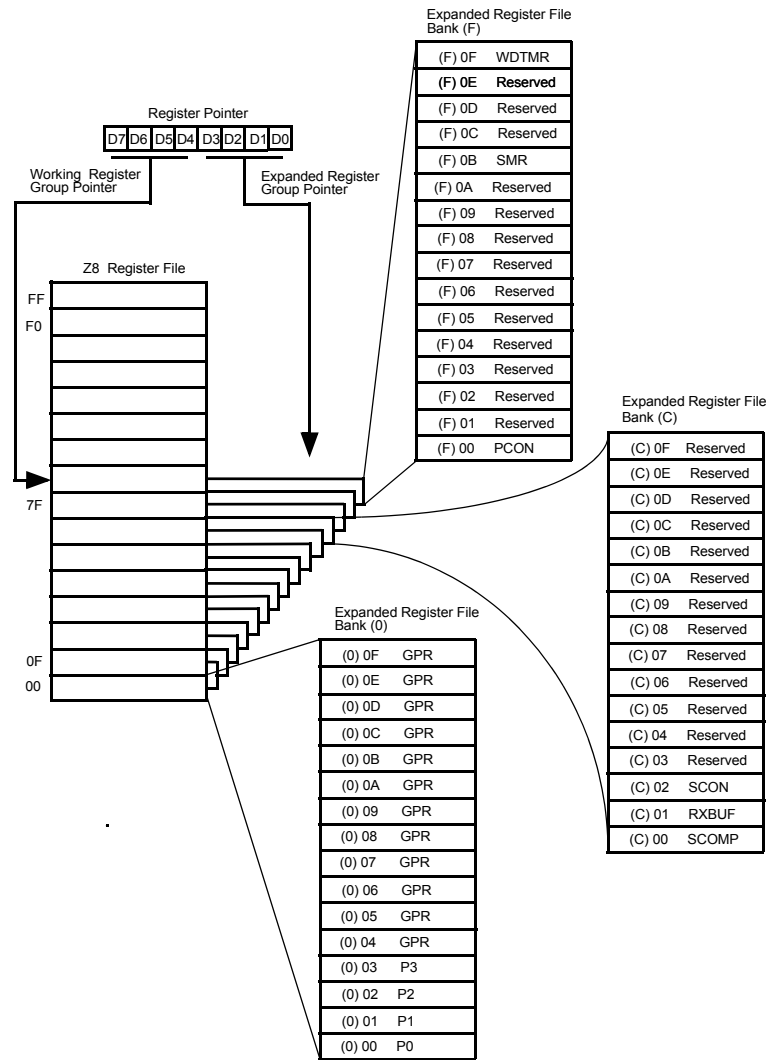
*Note: The fully implemented register file is shown. Refer to the specific product specification for actual register file architecture implemented.

Figure 6. Expanded Register File Architecture

Currently, three out of the possible sixteen Z8 ERF Banks have been implemented. ERF Bank 0, also known as the Z8® Standard Register File, has all 256 bytes defined, as displayed in Figure 7 on page 12. Only Working Register Group 0 (register addresses 00h to 0Fh) has been defined for ERF Bank C and ERF Bank F (see Table 4 on page 12). All other working register groups in ERF Banks C and F, as well as the remaining thirteen ERF Banks, are not implemented. All are reserved for future use.

When an ERF Bank is selected, register addresses `00h` to `0Fh` access those sixteen ERF Bank registers—in effect replacing the first sixteen locations of the Z8® Standard Register File.

For example, if ERF Bank C is selected, the Z8 Standard Registers `00h` through `0Fh` are no longer accessible. Registers `00h` through `0Fh` are now the 16 registers from ERF Bank C, Working Register Group 0. No other Z8 Standard Registers are affected because only Working Register Group 0 is implemented in ERF Bank C.

Access to the ERF is accomplished through the Register Pointer (`FDh`). The lower nibble of the Register Pointer determines the ERF Bank while the upper nibble determines the Working Register Group within the register file, as displayed in Figure 7.

Select ERF Bank Ch
Working Register Group 7h
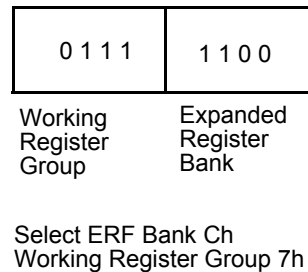
Figure 7. Register Pointer Example

The value of the lower nibble in the Register Pointer (`FDh`) corresponds to the ERF Bank identification. Table 4 lists the lower nibble value and the register file assigned to it.

Table 4. ERF Bank Address

Register Pointer (FDh) Low Nibble	Hex	Register File
0000b	0	Z8 Standard Register File*
0001b	1	Expanded Register File Bank 1
0010b	2	Expanded Register File Bank 2
0011b	3	Expanded Register File Bank 3
0100b	4	Expanded Register File Bank 4
0101b	5	Expanded Register File Bank 5
0110b	6	Expanded Register File Bank 6
0111b	7	Expanded Register File Bank 7
1000b	8	Expanded Register File Bank 8

Table 4. ERF Bank Address (Continued)

Register Pointer (FDh) Low Nibble	Hex	Register File
1001b	9	Expanded Register File Bank 9
1010b	A	Expanded Register File Bank A
1011b	B	Expanded Register File Bank B
1100b	C	Expanded Register File Bank C
1101b	D	Expanded Register File Bank D
1110b	E	Expanded Register File Bank E
1111b	F	Expanded Register File Bank F

*The Z8® Standard Register File is equivalent to Expanded Register File Bank 0.

The upper nibble of the register pointer selects which group of 16 bytes in the Register File, out of the 256 total bytes, is accessed as working registers. Table 5 lists an example.

Table 5. Register Pointer Access Example

R253 RP = 00h	;ERF Bank 0, Working Reg. Group 0
	R0 = Port 0 = 00h
	R1 = Port 1 = 01h
	R2 = Port 2 = 02h
	R3 = Port 3 = 03h
	R11 = GPR 0Bh
	R15 = GPR 0Fh
If R253 RP = 0Fh	;ERF Bank F, Working Reg. Group 0
	R0 = PCON = 00h
	R1 = Reserved = 01h
	R2 = Reserved = 02h
	R11 = SMR = 0Bh
	R15 = WDTMR = 0Fh

Table 5. Register Pointer Access Example (Continued)

If R253 RP = FFh	
	;ERF Bank F, Working Reg. Group F.
	00h = PCON
R0 = SI0	01h = Reserved
R1 = TMR	02h = Reserved
	...
R2 = T1	0Bh = SMR
	...
R15 = SPL	0Fh = WDTMR

Because enabling an ERF Bank (C or F) only changes register addresses 00h to 0Fh, the working register pointer can be used to access either the selected ERF Bank (Bank C or F, Working Register Group 0) or the Z8® Standard Register File (ERF Bank 0, Working Register Groups 1 through F).

When an ERF Bank other than Bank 0 is enabled, the first 16 bytes of the Z8 Standard Register File (I/O ports 0 to 3, Groups 4 to F) are no longer accessible (the selected ERF Bank, Registers 00h to 0Fh are accessed instead). It is important to re-initialize the Register Pointer to enable ERF Bank 0 when these registers are required for use.

The SPI register is mapped into ERF Bank C. Access is easily done using the example in Table 6.

Table 6. ERF Bank C Access Example

LD	RP, #0Ch	;Select ERF Bank C working
		;register group 0 for access.
LD	R2,#xx	;access SCON
LD	R1, #xx	;access RXBUF
LD	RP, #00h	;Select ERF Bank 0 so I/O ports
		;are again accessible.

Table 7. Z8 ERF Bank Layout

ERF Bank	ERF
Fh	`PCON, SMR, WDT, (00h, 0Bh, 0Fh),` Working Register Group 0 only implemented.
Eh	Not implemented (reserved)
Dh	Not implemented (reserved)
Ch	SPI Registers: `SCOMP, RXBUF, SCON (00h, 01h, 02h),` Working Register Group 0 only implemented.
Bh	Not implemented (reserved)
Ah	Not implemented (reserved)
9h	Not implemented (reserved)
8h	Not implemented (reserved)
7h	Not implemented (reserved)
6h	Not implemented (reserved)
5h	Not implemented (reserved)
4h	Not implemented (reserved)
3h	Not implemented (reserved)
2h	Not implemented (reserved)
1h	Not implemented (reserved)
0h	Z8 Ports 0, 1, 2, 3, and General-Purpose Registers `04h` to `EFh`, and control registers `F0h` to `FFh`.

Refer to the specific product specification to determine the above registers are implemented.

Z8® Control and Peripheral Registers

Standard Z8 Registers

The standard Z8 control registers govern the operation of the CPU. Any instruction which references the register file can access these control registers. The following control registers are available:

- Interrupt Priority Register (IPR)

- Interrupt Mask Register (IMR)

- Interrupt Request Register (IRQ)

- Program Control Flags (FLAGS)

- Register Pointer (RP)

- Stack Pointer High-Byte (SPH)

- Stack Pointer Low-Byte (SPL)

The Z8® CPU uses a 16-bit Program Counter (PC) to determine the sequence of current program instructions. The PC is not an addressable register.

Peripheral registers are used to transfer data, configure the operating mode, and control the operation of the on-chip peripherals. Any instruction that references the register file can access the peripheral registers. The peripheral registers are:

- Serial I/O (SIO)

- Timer Mode (TMR)

- Timer/Counter 0 (T0)

- T0 Prescaler (PRE0)

- Timer/Counter 1 (T1)

- T1 Prescaler (PRE1)

- Port 0–1 Mode (P01M)

- Port 2 Mode (P2M)

- Port 3 Mode (P3M)

In addition, the four port registers (P0–P3) are considered to be peripheral registers.

Expanded Z8 Registers

The expanded Z8 control registers govern the operation of additional features or peripherals. Any instruction which references the register file can access these registers.

The ERF contains the control registers for WDT, Port Control, Serial Peripheral Interface (SPI), and the SMR functions. Figure 6 on page 11 displays the layout of the Register Banks in the ERF. Register Bank C in the ERF consists of the registers for the SPI. Table 8 lists the registers within ERF Bank C, Working Register Group 0.

Table 8. ERF Bank C WR Group 0

Register	Function	Working Register
F	Reserved	R15
E	Reserved	R14
D	Reserved	R13

Table 8. ERF Bank C WR Group 0 (Continued)

Register	Function	Working Register
C	Reserved	R12
B	Reserved	R11
A	Reserved	R10
9	Reserved	R9
8	Reserved	R8
7	Reserved	R7
6	Reserved	R6
5	Reserved	R5
4	Reserved	R4
3	Reserved	R3
2	SPI Control (SCON)	R2
1	SPI Tx/Rx Data (Roxburgh)	R1
0	SPI Compare (SCOMP)	R0

Working Register Group 0 in ERF Bank 0 consists of the registers for Z8® General-Purpose Registers and ports. Table 9 lists the registers within this group.

Table 9. ERF Bank 0 WR Group 0

Register	Function	Working Register
F	General-Purpose Register	R15
E	General-Purpose Register	R14
D	General-Purpose Register	R13
C	General-Purpose Register	R12
B	General-Purpose Register	R11
A	General-Purpose Register	R10
9	General-Purpose Register	R9
8	General-Purpose Register	R8
7	General-Purpose Register	R7
6	General-Purpose Register	R6
5	General-Purpose Register	R5

Table 9. ERF Bank 0 WR Group 0 (Continued)

Register	Function	Working Register
4	General-Purpose Register	R4
3	Port 3	R3
2	Port 2	R2
1	Port 1	R1
0	Port 0	R0

Working Register Group 0 in ERF Bank F consists of the control registers for STOP mode, WDT, and port control. Table 10 lists the registers within this group.

Table 10. ERF Bank F WR Group 0

Register	Function	Working Register
F	WDTMR	R15
E	Reserved	R14
D	Reserved	R13
C	Reserved	R12
B	SMR	R11
A	Reserved	R10
9	Reserved	R9
8	Reserved	R8
7	Reserved	R7
6	Reserved	R6
5	Reserved	R5
4	Reserved	R4
3	Reserved	R3
2	Reserved	R2
1	Reserved	R1
0	PCON	R0

Program Memory

The first 12 bytes of Program Memory are reserved for the interrupt vectors, as displayed in Figure 8 on page 20. These locations contain six 16-bit vectors that correspond to the six available interrupts. Address 12 up to the maximum ROM address consists of on-chip mask-programmable ROM. Refer to the product data sheet for the exact program, data, register memory size, and address range available. At addresses outside the internal ROM, the Z8® CPU executes external Program Memory fetches through Port 0 and Port 1 in Address/Data mode for devices with Port 0 and Port 1 featured. Otherwise, the program counter continues to execute NOPs up to address `FFFFh`, roll over to `0000h`, and continue to fetch executable code (see Figure 8).

The internal Program Memory is one-time programmable (OTP) or mask programmable dependent on the specific device. A ROM protect feature prevents dumping of the ROM contents by inhibiting execution of the LDC, LDCI, LDE, and LDEI instructions to Program Memory in all modes. ROM look-up tables cannot be used with this feature.

The ROM Protect option is mask-programmable, to be selected when the ROM code is submitted. For the OTP ROM, the ROM Protect option is an OTP programming option.
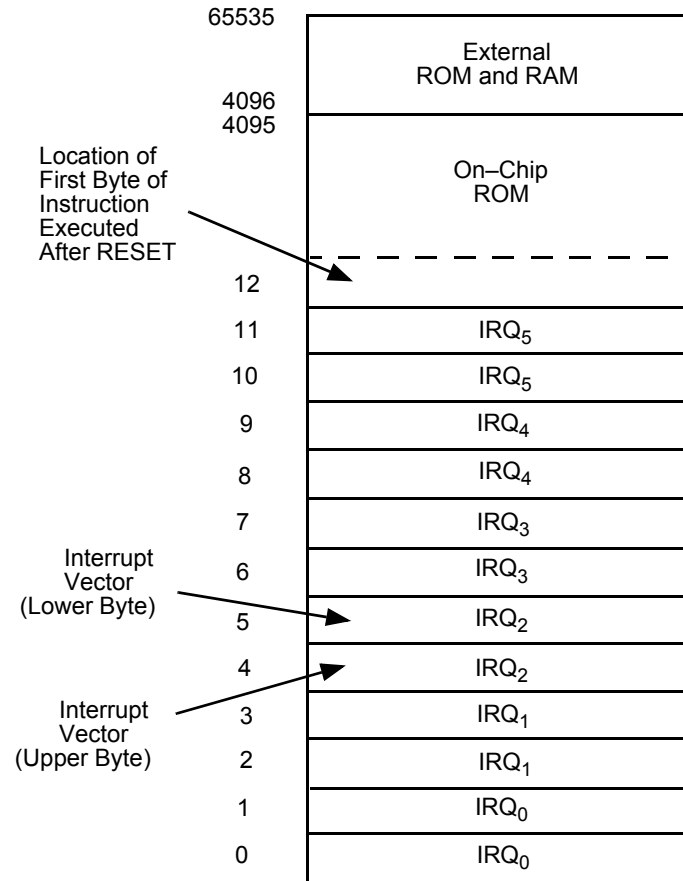
Figure 8. Z8® Program Memory Map

Z8® External Memory

Z8 CPU, in some cases, has the capability to access external Program Memory with the 16-bit Program Counter. To access external Program Memory the Z8 CPU offers multi-plexed address/data lines (AD7–AD0) on Port 1 and address lines (A15–A8) on Port 0. This feature only applies to devices that offer Port 0 and Port 1. The maximum external address is FFFF. This memory interface is supported by the control lines $\overline{AS}$ (Address Strobe), $\overline{DS}$ (Data Strobe), and R/W (Read/Write). The origin of the external Program Memory starts after the last address of the internal ROM. displays an example of external Program Memory for the Z8 CPU.

External Data Memory

The Z8 CPU, in some cases, can address up to 60 KB of external data memory beginning at location 4096. External data memory ($\overline{DM}$) can be included with, or separated from, the external Program Memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and Program Memory space. The state of the $\overline{DM}$ signal is controlled by the type of instruction being executed. An LDC opcode references Program Memory ($\overline{DM}$ inactive), and an LDE instruction references data memory ($\overline{DM}$ active Low). You must configure Port 3 Mode Register (P3M) bits D3 and D4 for this mode.
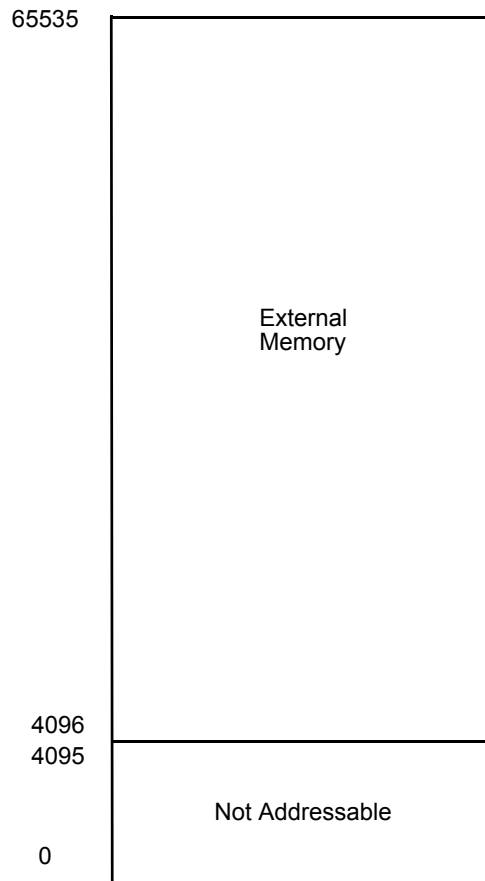
*Note: For additional information on using external memory, see Chapter 10 of this manual. For exact memory addressing options available, see the device product specification.

Figure 9. External Memory Map

Z8® Stacks

Stack operations can occur in either the Z8 Standard Register File or external data memory. Under software control, Port 0–1 Mode register (`F8h`) selects the stack location. Only the General-Purpose Registers can be used for the stack when the internal stack is selected.

The register pair `FEh` and `FFh` form the 16-bit Stack Pointer (SP), that is used for all stack operations. The stack address is stored with the MSB in `FEh` and LSB in `FFh`, see Figure 10.

FFh | LOWER Byte | Stack Pointer Low

FEh | UPPER Byte | Stack Pointer High
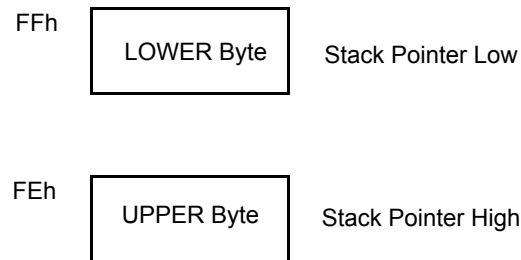
Figure 10. Stack Pointer

The stack address is decremented prior to a PUSH operation and incremented after a POP operation. The stack address always points to the data stored on the top of the stack. The Z8 CPU stack is a return stack for CALL instructions and interrupts, as well as a data stack.

During a CALL instruction, the contents of the PC are saved on the stack. The PC is restored during a RETURN instruction. Interrupts cause the contents of the PC and Flag registers to be saved on the stack. The IRET instruction restores them (see Figure 11 on page 23).

When the Z8 CPU is configured for an internal stack (using the Z8 Standard Register File), register `FFh` serves as the Stack Pointer. The value in `FEh` is ignored. `FEh` can be used as a general-purpose register in this case only.

An overflow or underflow can occur when the stack address is incremented or decremented during normal stack operations. The programmer must prevent this occurrence, or unpredictable operation happens.
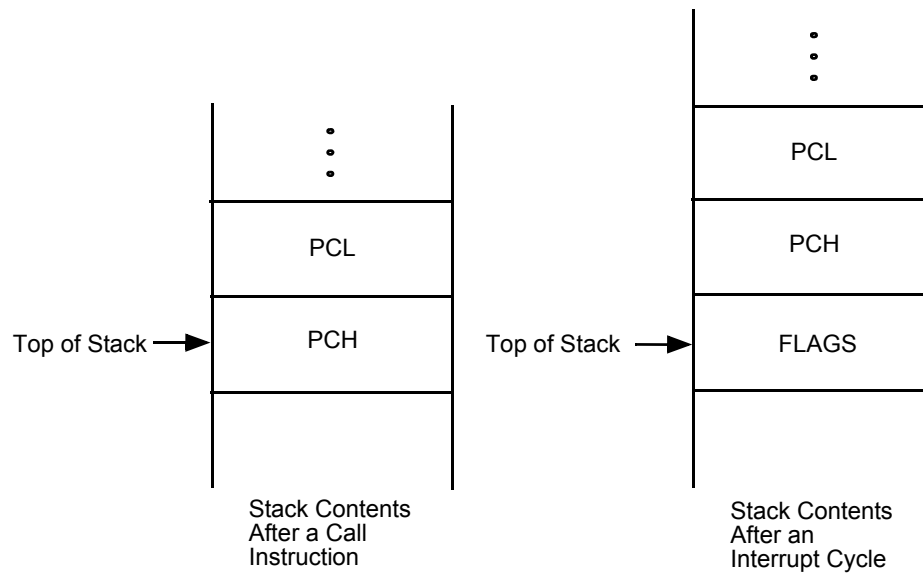
Figure 11. Stack Operations

Clock

Z8® CPU derives its timing from on-board clock circuitry connected to pins XTAL1 and XTAL2. The clock circuitry consists of an oscillator, a divide-by-two shaping circuit, and a clock buffer. Figure 12 displays the clock circuitry. The oscillator's input is XTAL1 and its output is XTAL2. The clock can be driven by a crystal, a ceramic resonator, LC clock, RC, or an external clock source.

Frequency Control

In some cases, the Z8 CPU has an EPROM/OTP option or a Mask ROM option bit to bypass the divide-by-two flip flop in Figure 12. This feature is used in conjunction with the low EMI option. When low EMI is selected, the device output drive and oscillator drive is reduced to approximately 25 percent of the standard drive and the divide-by-two flip flop is bypassed such that the XTAL clock frequency is equal to the internal system clock frequency. In this mode, the maximum frequency of the XTAL clock is 4 MHz. Refer to specific product specification for availability of options and output drive characteristics.
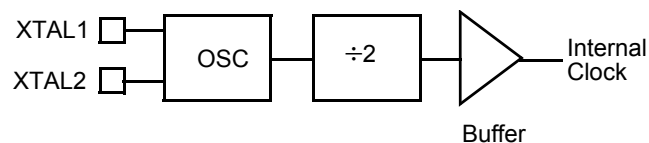
Figure 12. Z8® CPU Clock Circuit

Clock Control

In some cases, the Z8 CPU offers software control of the internal system clock via programming register bits. The bits are located in the Stop Mode Recovery Register in Expanded Register File Bank F, Register `0Bh`. This register selects the clock divide value and determines the mode of Stop Mode Recovery (see Figure 13 on page 25). Refer to the specific product specification for availability of this feature/register.

```
SMR (F) OB
┌──┬──┬──┬──┬──┬──┬──┬──┐
│D7│D6│D5│D4│D3│D2│D1│D0│
└──┴──┴──┴──┴──┴──┴──┴──┘
```

SCLK ÷ TCLK Divide by 16
0 OFF **
1 ON

External Clock Divide Mode by 2
0 = SCLK ÷ TCLK = XTAL ÷ 2*
1 = SCLK ÷ TCLK = XTAL

* Default setting after RESET.

**Default setting after RESET and Stop Mode Recovery.
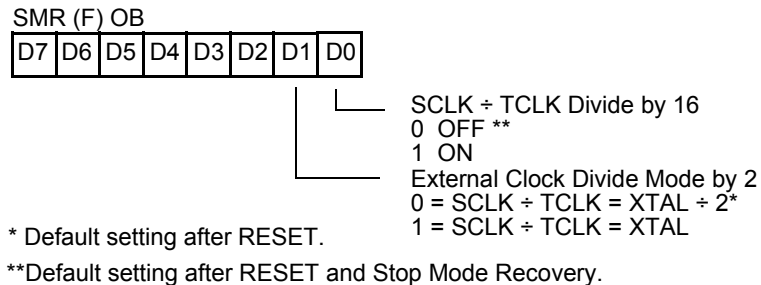
Figure 13. Stop Mode Recovery Register (Write-Only Except D7, Which is Read-Only)

SCLK ÷ TCLK Divide-By-16 Select

The D0 bit of the SMR controls a divide-by-16 prescaler of SCLK ÷ TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

External Clock Divide-By-Two

The D1 bit can eliminate the oscillator divide-by-two circuitry. When this bit is 0, SCLK (System Clock) and TCLK (Timer Clock) are equal to the external clock frequency divided by two. The SCLK ÷ TCLK is equal to the external clock frequency when this bit is set (D1 = 1). Using this bit, together with D7 of PCON, further helps lower EMI (D7 (PCON) = 0, D1 (SMR) = 1). The default setting is 0. Maximum frequency is 4 MHz with D1 = 1 (see Figure 14 on page 26).
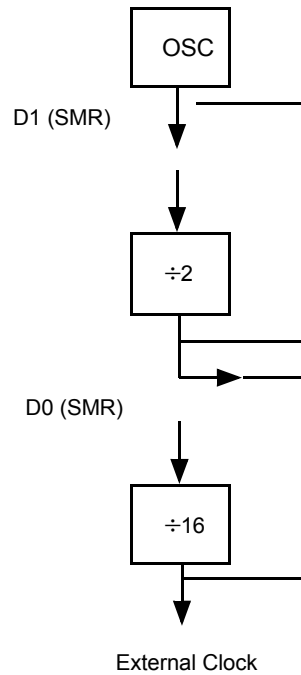
Figure 14. External Clock Circuit

Oscillator Control

In some cases, the Z8® CPU offers software control of the oscillator to select low EMI drive or standard drive. The selection is done by programming bit D7 of the Port Configuration (PCON) register (see Figure 15 on page 26). The PCON register is located in Expanded Register File Bank F, Register 00h.

A 1 in bit D7 configures the oscillator with standard drive, while a 0 configures the oscillator with Low EMI drive. This only affects the drive capability of the oscillator and does not affect the relationship of the XTAL clock frequency to the internal system clock (SCLK).
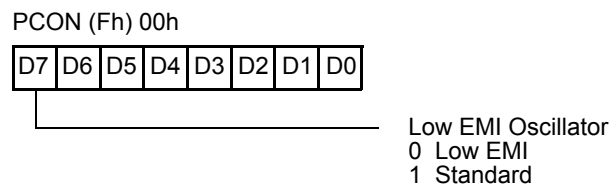
Figure 15. Port Configuration Register (Write-Only)

Oscillator Operation

The Z8® CPU uses a Pierce oscillator with an internal feedback (see Figure 16). The advantages of this circuit are low cost, large output signal, low-power level in the crystal, stability with respect to V_{CC} and temperature, and low impedances (not disturbed by stray affects).

One drawback is the requirement for high gain in the amplifier to compensate for feedback path losses. The oscillator amplifies its own noise at start-up until it settles at the frequency that satisfies the gain/phase requirements A x B = 1, where $A = V_0/V_I$ is the gain of the amplifier and $B = V_I/V_0$ is the gain of the feedback element. The total phase shift around the loop is forced to zero (360 degrees). Because VIN must be in phase with itself, the amplifier/inverter provides 180 degree phase shift and the feedback element is forced to provide the other 180 degrees of phase shift.

R1 is a resistive component placed from output to input of the amplifier. The purpose of this feedback is to bias the amplifier in its linear region and to provide the start-up transition.

Capacitor C_2 combined with the amplifier output resistance provides a small phase shift. It also provides some attenuation of overtones.

Capacitor C_1 combined with the crystal resistance provides additional phase shift.

C_1 and C_2 can affect the start-up time if they increase dramatically in size. As C_1 and C_2 increase, the start-up time increases until the oscillator reaches a point where it does not start up any more.

For fast and reliable oscillator start-up over the manufacturing process range, Zilog® recommends that the load capacitors be sized as low as possible without resulting in overtone operation.
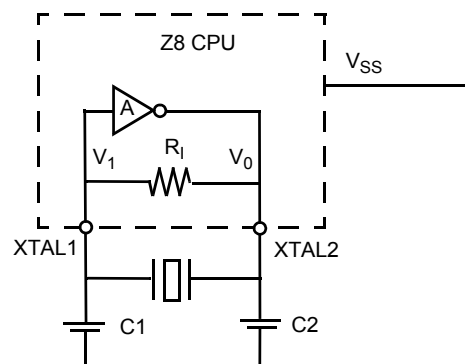
Figure 16. Pierce Oscillator with Internal Feedback Circuit

Layout

Traces connecting crystal, caps, and the Z8® CPU oscillator pins should be as short and wide as possible. This reduces parasitic inductance and resistance. The components (caps, crystal, resistors) should be placed as close as possible to the oscillator pins of the Z8 CPU.

The traces from the oscillator pins of the IC and the ground side of the lead caps should be guarded from all other traces (clock, V_{CC}, address/data lines, system ground) to reduce cross talk and noise injection. This is usually accomplished by keeping other traces and system ground trace planes away from the oscillator circuit and by placing a Z8 CPU device V_{SS} ground ring around the traces/components. The ground side of the oscillator lead caps should be connected to a single trace to the Z8 CPU's V_{SS} (GND) pin. It should not be shared with any other system ground trace or components except at the Z8 CPU's V_{SS} pin. This is to prevent differential system ground noise injection into the oscillator (see).

Indications of an Unreliable Design

Start-up time and output level are two major indicators that are used in working designs to determine their reliability over full lot and temperature variations. These two indicators are described below.

Start-Up Time—If start-up time is excessive, or varies widely from unit to unit, there is probably a gain problem. C1/C2 must be reduced; the amplifier gain is not adequate at frequency, or crystal resistance is too large.

Output Level—The signal at the amplifier output should swing from ground to V_{CC}. This indicates there is adequate gain in the amplifier. As the oscillator starts up, the signal amplitude grows until clipping occurs, at which point the loop gain is effectively reduced to unity and constant oscillation is achieved. A signal of less than 2.5 V peak-to-peak is an indication that low gain may be a problem. Either C_1 or C_2 should be made smaller or a low-resistance crystal should be used.

Circuit Board Design Rules

The following circuit board design rules are suggested:

- To prevent induced noise the crystal and load capacitors should be physically located as close to the Z8 CPU as possible.

- Signal lines should not run parallel to the clock oscillator inputs. In particular, the crystal input circuitry and the internal system clock output should be separated as much as possible.

- V_{CC} power lines should be separated from the clock oscillator input circuitry.

- Resistivity between XTAL1 or XTAL2 and the other pins should be greater than 10 MΩ.
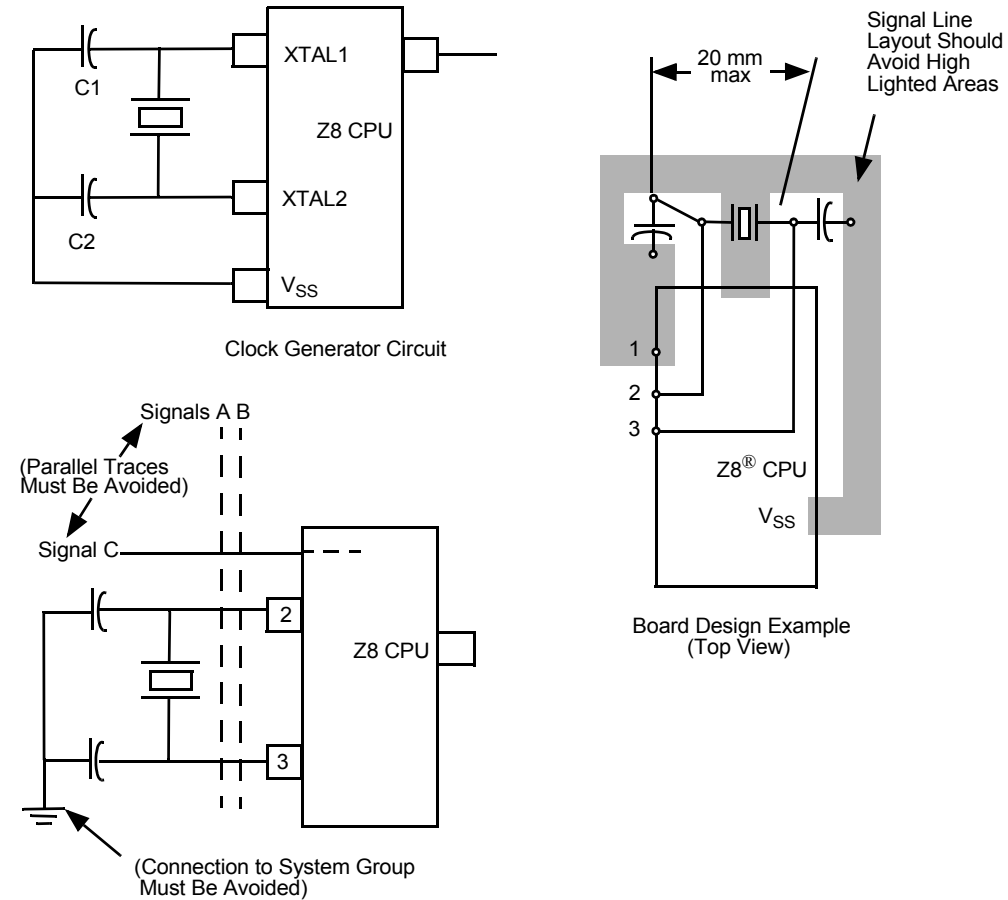
Figure 17. Circuit Board Design Rules

Crystals and Resonators

Crystals and ceramic resonators, displayed in Figure 18 should have the characteristics listed in Table 11 to ensure proper oscillator operation.

Table 11. Crystal/Resonator Characteristics

Crystal Cut	AT (crystal only)
Mode	Parallel, Fundamental mode
Crystal Capacitance	<7 pF
Load Capacitance	10 pF < CL < 220 pF, 15 typical
Resistance	100Ω max

Depending on operation frequency, the oscillator may require the addition of capacitors C1 and C2 (displayed in Figure 18). The capacitance values are dependent on the manufacturer's crystal specifications.
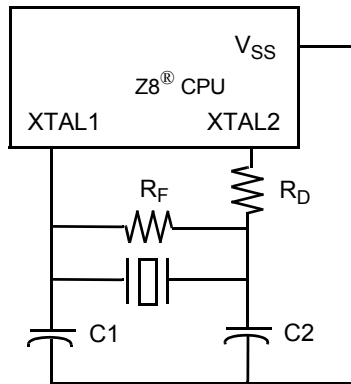
Figure 18. Crystal/Ceramic Resonator Oscillator
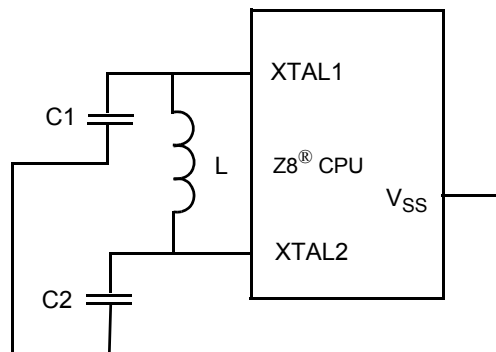
Figure 19. LC Clock

In most cases, the R_D is 0Ω and R_F is infinite. It is determined and specified by the crystal/ceramic resonator manufacturer. The R_D can be increased to decrease the amount of drive from the oscillator output to the crystal. It can also be used as an adjustment to avoid clipping of the oscillator signal to reduce noise. The R_F can be used to improve the start-up of the crystal/ceramic resonator. The Z8® oscillator already has an internal shunt resistor in parallel to the crystal/ceramic resonator.
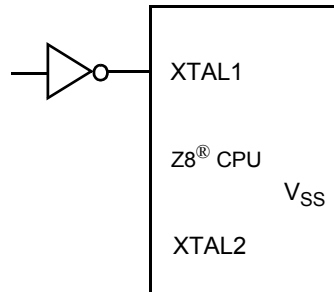
Figure 20. External Clock

In Figure 18 through Figure 20, Zilog® recommends that you connect the load capacitor ground trace directly to the V_{SS} (GND) pin of the Z8® CPU to ensure that no system noise is injected into the Z8 clock. This trace should not be shared with any other components except at the V_{SS} pin of the Z8 CPU.

In some cases, the Z8 CPU's XTAL1 pin also functions as one of the EPROM high-voltage mode programming pins or as a special factory test pin. In this case, applying 2 V above V_{CC} on the XTAL1 pin causes the device to enter one of these modes. Because this pin accepts high voltages to enter these respective modes, the standard input protection diode to V_{CC} is not on XTAL1. Zilog recommends that in applications where the Z8 CPU is exposed to much system noise, a diode from XTAL1 to V_{CC} be used to prevent accidental enabling of these modes. This diode does not affect the crystal/ceramic resonator operation.

Note: *A parallel resonant crystal or resonator data sheet specifies a load capacitor value that is the series combination of C_1 and C_2, including all parasitics (PCB and holder).*

LC Oscillator

The Z8 CPU oscillator can use a LC network to generate a XTAL clock (see Figure 19 on page 30).

The frequency stays stable over V_{CC} and temperature. The oscillation frequency is determined by following equation.

$$\text{Frequency} = \frac{1}{2\pi(LC_T)^{1/2}}$$

where L is the total inductance including parasitics and C_T is the total series capacitance including the parasitics.

Simple series capacitance is calculated using the following equation:

$$\frac{1}{C_T} = \frac{1}{C_1} + \frac{1}{C_2}$$

$$\text{If } C_1 = C_2$$

$$\frac{1}{C_T} = \frac{2}{C_1}$$

$$C_1 = 2C_T$$

Sample calculation for capacitance C_1 and C_2 of 5.83 MHz frequency and inductance value of 27 μH.

$$5.83\ (10^6) = \frac{1}{2\pi\ [2.7\ (10^{-6})\ C_T]\ 1/2}$$

$C_T = 27.6$ pF

Therefore, $C_1 = 55.2$ pF and $C_2 = 55.2$ pF.

RC Oscillator

In some cases, the Z8® CPU features an RC oscillator option. Refer to the specific product specification for availability. The RC oscillator requires a resistor across XTAL1 and XTAL2. An additional load capacitor is required from the XTAL1 input to V_{SS} pin, see Figure 21.
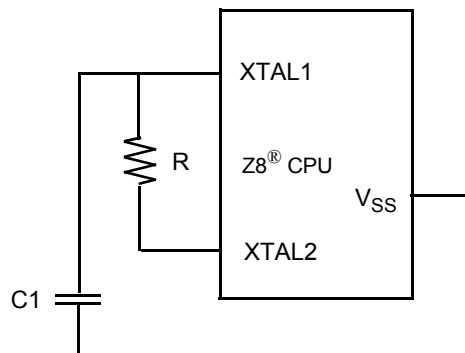
Figure 21. RC Clock

Reset

This section describes the Z8® CPU reset conditions, reset timing, and register initialization procedures. Reset is generated by Power-On Reset (POR), Reset Pin, Watchdog Timer (WDT), and Stop Mode Recovery.

A system reset overrides all other operating conditions and puts the Z8 CPU into a known state. To initialize the chip's internal logic, the $\overline{RESET}$ input must be held Low for at least 21 SCP or 5 XTAL clock cycles. The control register and ports are reset to their default conditions after a POR, a reset from the $\overline{RESET}$ pin, or WDT time-out while in RUN mode and HALT mode. The control registers and ports are not reset to their default conditions after Stop Mode Recovery and WDT time-out while in STOP mode.

While $\overline{RESET}$ pin is Low, $\overline{AS}$ is output at the internal clock rate, $\overline{DS}$ is forced Low, and R/$\overline{W}$ remains High. The program counter is loaded with `000Ch`. I/O ports and control registers are configured to their default reset state.

Resetting the Z8 CPU does not affect the contents of the general-purpose registers.

Reset Pin, Internal POR Operation

In some cases, the Z8 CPU hardware $\overline{RESET}$ pin initializes the control and peripheral registers, as listed in Table 12 on page 34 through Table 15 on page 37. Specific reset values are shown by 1 or 0, while bits whose states are unknown are indicated by the letter U. Table 12 on page 34 through Table 15 on page 37 list the reset conditions for the Z8 CPU.

▶ **Note:** *The register file reset state is device dependent. Refer to the selected device product specifications for register availability and reset state.*

Table 12. Sample Control and Peripheral Register Reset Values (ERF Bank 0)

Register (Hex)	Register Name	Bits								Comments
		7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
F0	Serial I/O	U	U	U	U	U	U	U	U	
F1	Timer Mode	0	0	0	0	0	0	0	0	Counter/Timers stopped.
F2	Counter/Timer1	U	U	U	U	U	U	U	U	
F3	T1 Prescaler	U	U	U	U	U	U	0	0	Single-pass count mode, external clock source.
F4	Counter/Timer0	U	U	U	U	U	U	U	U	
F5	T0 Prescaler	U	U	U	U	U	U	U	0	Single-pass count mode.
F6	Port 2 Mode	1	1	1	1	1	1	1	1	All inputs.
F7	Port 3 Mode	0	0	0	0	0	0	0	0	Port 2 open-drain, P33–P30 Input, P37–P34 Output.
F8	Port 0–1 Mode	0	1	0	0	1	1	0	1	Internal Stack, Normal Memory Timing.
F9	Interrupt Priority	U	U	U	U	U	U	U	U	
FA	Interrupt Request	0	0	0	0	0	0	0	0	All Interrupts Cleared.
FB	Interrupt Mask	0	U	U	U	U	U	U	U	Interrupts Disabled.
FC	Flags	U	U	U	U	U	U	U	U	
FD	Register Pointer	0	0	0	0	0	0	0	0	
FE	Stack Pointer (High)	U	U	U	U	U	U	U	U	
FF	Stack Pointer (Low)	U	U	U	U	U	U	U	U	

Program execution starts 5 to 10 clock cycles after internal $\overline{\text{RESET}}$ has returned High. The initial instruction fetch is from location `000Ch`. Figure 22 on page 35 displays reset timing.
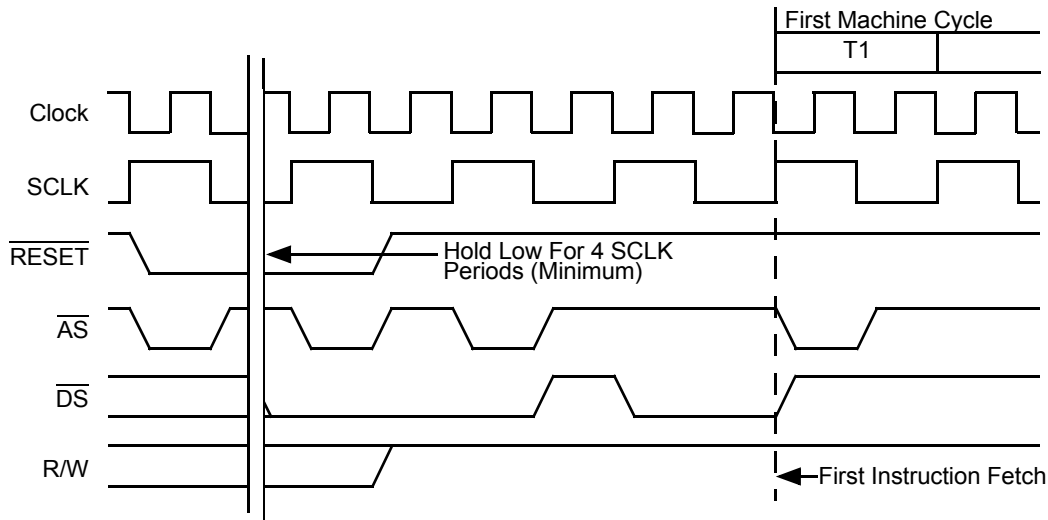
Figure 22. Reset Timing

After a reset, the first routine executed should be one that initializes the control registers to the required system configuration.

The $\overline{\text{RESET}}$ pin is the input of a Schmitt-Triggered circuit. Resetting the Z8® CPU initializes port and control registers to their default states. To form the internal reset line, the output of the trigger is synchronized with the internal clock. The clock must therefore be running for $\overline{\text{RESET}}$ to function. It requires 4 internal system clocks after reset is detected for the Z8 CPU to reset the internal circuitry. An internal pull-up, combined with an external capacitor of 1 μF, provides enough time to properly reset the Z8 CPU (see Figure 23 on page 36). In some cases, the Z8 CPU has an internal POR timer circuit that holds the Z8 CPU in reset mode for a duration (T_{POR}) before releasing the device out of reset. On these Z8 devices, the internally generated reset drives the reset pin low for the POR time. Any devices driving the reset line must be open-drained in order to avoid damage from possible conflict during reset conditions. This reset time allows the on-board clock oscillator to stabilize.

To avoid asynchronous and noisy reset problems, the Z8 CPU is equipped with a reset filter of four external clocks (4TpC). If the external reset signal is less than 4TpC in duration, no reset occurs. On the fifth clock after the reset is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external reset, whichever is longer. During the reset cycle, $\overline{\text{DS}}$ is held active low while $\overline{\text{AS}}$ cycles at a rate of the internal system clock. Program execution begins at location `000Ch`, 5-10 TpC cycles after $\overline{\text{RESET}}$ is released. For the internal Power-On Reset, the reset output time is specified as T_{POR}. Refer to specific product specifications for actual values.
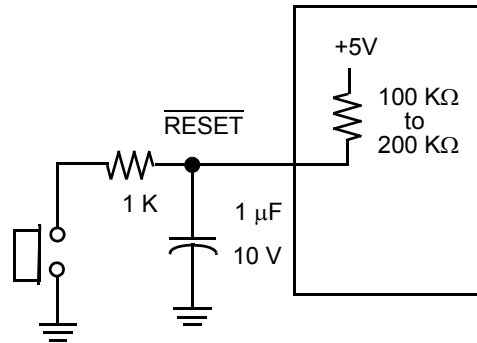
Figure 23. Example of External Power-On Reset Circuit

Table 13. ERF Bank 0 Reset Values at RESET

Register (Hex)	Register Name	Bits								Comments
		7	6	5	4	3	2	1	0	
00	Port 0	U	U	U	U	U	U	U	U	Input mode, output set to push–pull.
01	Port 1	U	U	U	U	U	U	U	U	Input mode, output set to push–pull.
02	Port 2	U	U	U	U	U	U	U	U	Input mode, output set to open drain.
03	Port 3	1	1	1	1	U	U	U	U	Standard digital input and output Z86L7X Family Device Port P34-P37 = 0 (Except Z86L70/71/75) All other Z8 = 1.
04–EF	General-Purpose Registers 04h–EFh	U	U	U	U	U	U	U	U	Undefined.

Table 14. Sample Expanded Register File Bank C Reset Values

Register (Hex)	Register Name	Bits								Comments
		7	6	5	4	3	2	1	0	
00	SPI Compare (SCOMP)	0	0	0	0	0	0	0	0	
01	Receive Buffer (RxBUF)	U	U	U	U	U	U	U	U	
02	SPI Control (SCON)	U	U	U	U	0	0	0	0	

Table 15. Sample Expanded Register File Bank F Reset Values

| Register (Hex) | Register Name | **Bits** | | | | | | | | Comments |
		7	6	5	4	3	2	1	0	
00	Port Configuration (PCON)	1	1	1	1	1	1	1	0	Comparator outputs disabled on Port 3.
										Port 0 and 1 output is push–pull.
										Port 0, 1, 2, 3, and oscillator with standard output drive.
0B	Stop Mode Recovery (SMR)	0	0	1	0	0	0	0	0	Clock divide by 16 off.
										XTAL divide by 2.
										POR and/OR External Reset.
										Stop delay on.
										Stop recovery level is low, STOP Flag is POR.
0F	Watchdog Timer Mode (WDTMR)	U	U	U	0	1	1	0	1	512 TPC for WDT time out, WDT runs during STOP.
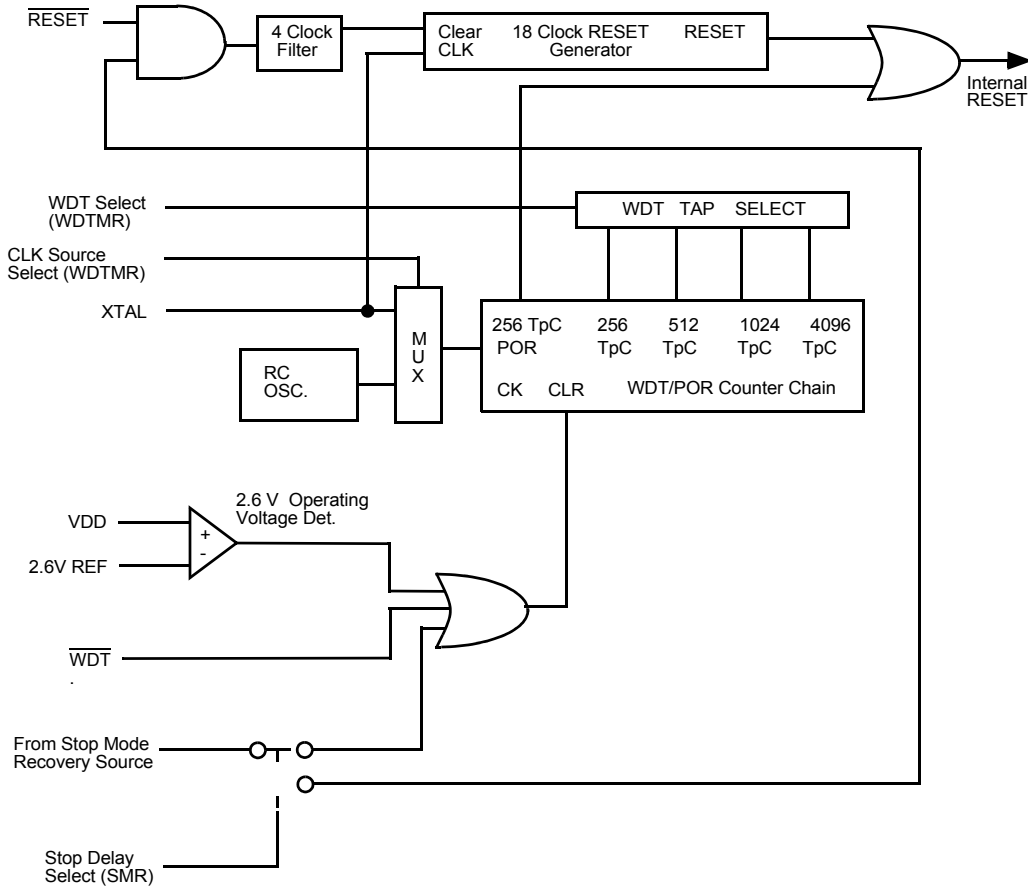
Figure 24. Example of Z8 Reset with RESET Pin, WDT, SMR, and POR
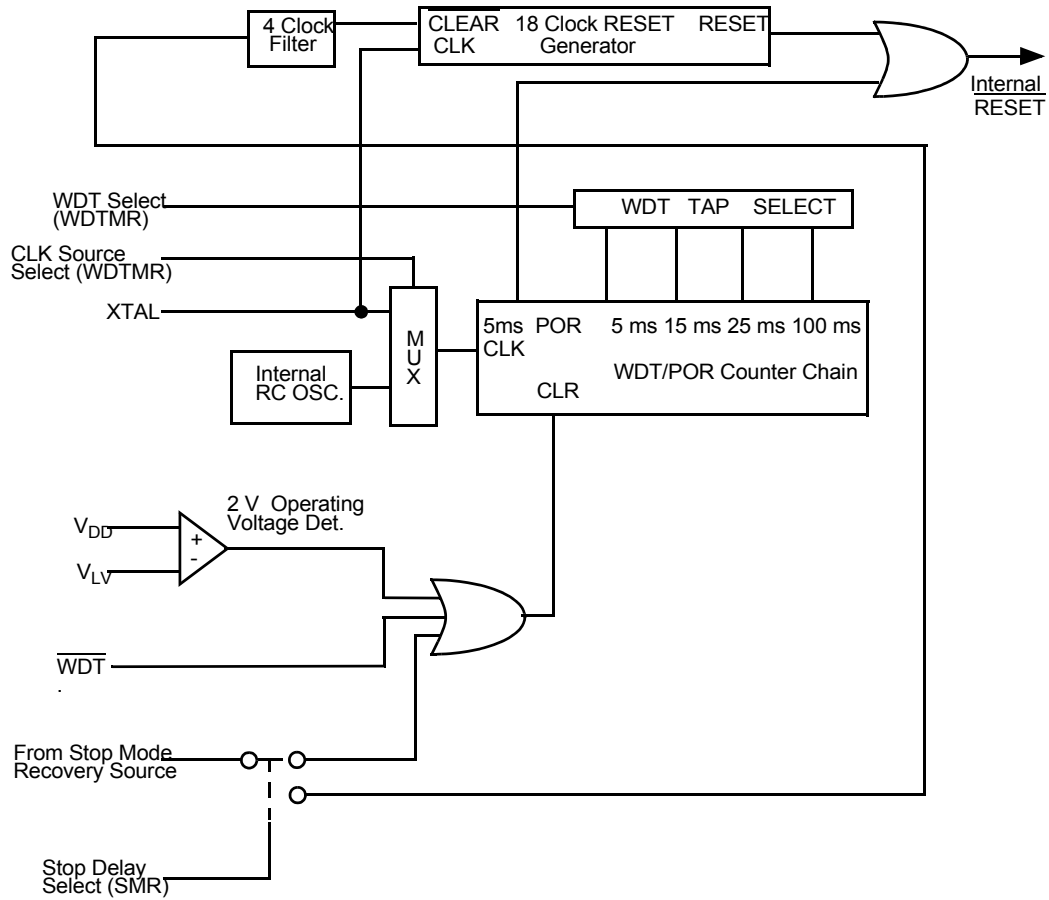
Figure 25. Example of Z8 Reset with WDT, SMR, and POR

Watchdog Timer

The WDT is a retriggerable one-shot timer that resets the Z8® CPU if it reaches its terminal count. When operating in the RUN or HALT modes, a WDT reset is functionally equivalent to a hardware POR reset. The WDT is initially enabled by executing the WDT instruction and refreshed on subsequent executions of the WDT instruction. The WDT cannot be disabled after it has been initially enabled. Permanently enabled WDTs are always enabled and the WDT instruction is used to refresh it. The WDT circuit is driven by an on-board RC oscillator or external oscillator from the XTAL1 pin. The POR clock source is selected with bit 4 of the Watchdog Timer Mode register (WDTMR). In some cases, a Z8 that offers the WDT but does not have a WDTMR register, has a fixed WDT time-out and uses the on board RC oscillator as the only clock source. Refer to specific product specifications for selectability of time-out, WDT during HALT and STOP modes, source of WDT clock, and availability of the permanently-on WDT option.

Execution of the WDT instruction affects the Z (zero), S (sign), and V (overflow) flags.

WDTMR (F) 0F

D7	D6	D5	D4	D3	D2	D1	D0

	INT	
WDT	RC	SYS
TAP*	OSC	CLK
00	5	128
01**	10	256
10	20	512
11	80	2048

WDT During HALT
  0 OFF
  1 ON *

WDT During STOP
  0 OFF
  1 ON *

XTAL1/INT RC
      Select for WDT
  0 On-Board RC *
  1 XTAL

Reserved (Must be 0)

* Must be 0 for Z86C03
** Default setting after RESET
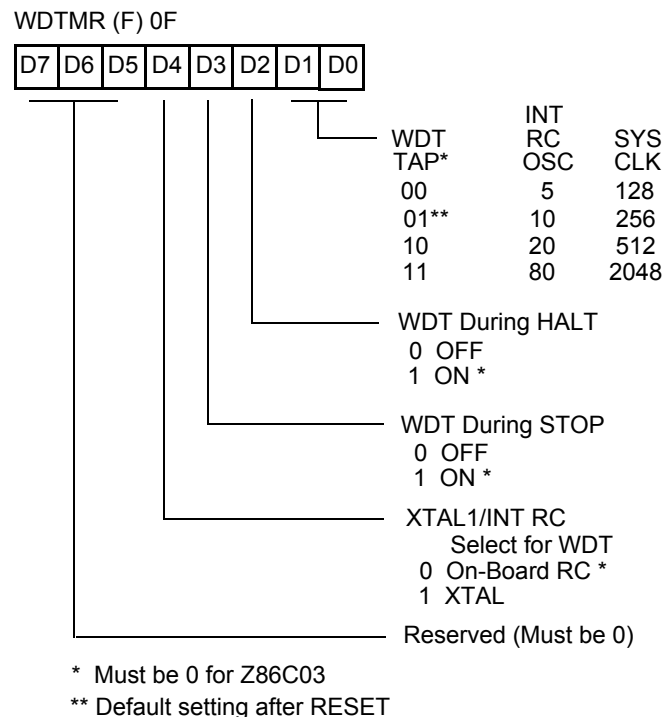
Figure 26. Example of Z8 Watchdog Timer Mode Register (Write-Only)

The WDTMR register is accessible only during the first 60 processor cycles from the execution of the first instruction after Power-On Reset, Watchdog Reset, or a Stop Mode Recovery. After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR is a write-only register.

WDTMR is located in ERF Bank F, register 0Fh. This register's control bits are described on the next two pages.

WDT Time Select—Bits D1 and D0 control a tap circuit that determines the time-out period. Table 16 on page 41 lists the different values that can be obtained. The default values of D1 and D0 are 0 and 1, respectively.

Table 16. Time-Out Period of the WDT

Time-Out of		Typical Time-Out of Internal RC	
D1	D0	OSC	System Clock
0	0	5 ms min	256 TpC
0	1	15 ms min	512 TpC
1	0	25 ms min	1024 TpC
1	1	100 ms min	4096 TpC

Notes: The values given are for V_{CC} = 5.0 V. See the device product specification for exact WDTMR time out select options available.
1. TpC = XTAL clock cycle
2. The default on reset is, D0 = 1 and D1 = 0.

WDT During HALT—The D2 bit determines whether or not the WDT is active during HALT mode. A 1 indicates active during HALT. The default is 1. A WDT time out during HALT mode resets control register ports to their default reset conditions.

WDT During STOP—The D3 bit determines whether or not the WDT is active during STOP mode. Because XTAL clock is stopped during STOP mode, unless as specified below, the on-board RC must be selected as the clock source to the POR counter. A 1 indicates active during STOP. The default is 1. If bits D3 and D4 are both set to 1, the WDT only, is driven by the external clock during STOP mode. This feature makes it possible to wake up from STOP mode from an internal source. Refer to specific product specifications for conditions of control and port registers when the Z8® CPU comes out of STOP mode. A WDT time out during STOP mode does not reset all control registers. The reset conditions of the ports from STOP mode due to WDT time out are the same as if recovered using any of the other STOP mode sources.

Clock Source for WDT—The D4 bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the internal RC oscillator.

Bits 5, 6, and 7—These bits are reserved.

V$_{CC}$ Voltage Comparator—An on-board voltage comparator checks that V$_{CC}$ is at the required level to insure correct operation of the device. Reset is globally driven if V$_{CC}$ is below the specified voltage. This feature is available in select ROM Z8 devices. See the device product specification for feature availability and operating range.

Power-On Reset

A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer function, T$_{POR}$. This POR time allows V$_{CC}$ and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

- Power fail to Power OK status (cold start)
- Stop Mode Recovery (if bit 5 of SMR = 1)
- WDT time-out

The POR time is specified as T$_{POR}$. On Z8 devices that feature a Stop Mode Recovery register (SMR), bit 5 selects whether the POR timer is used after Stop Mode Recovery or by-passed. If bit D5 = 1 then the POR timer is used. If bit 5 = 0 then the POR timer is by-passed. In this case, the Stop Mode Recovery source must be held in the recovery state for 5 T$_P$C or 5 crystal clocks to pass the reset signal internally. This option is used when the clock is provided with an RC/LC clock. See the device product specification for timing details.

POR (cold start) always resets the Z8® CPU control and port registers to their default condition. If a Z8 has a SMR, the warm start bit is reset to a 0 to indicate POR.
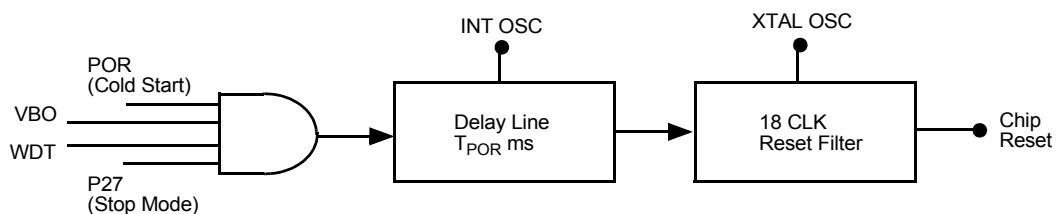
Figure 27. Example of Z8 with Simple SMR and POR

Input/Output Ports

Z8® CPU features up to 32 lines dedicated to input and output. These lines are grouped into four 8-bit ports known as Port 0, Port 1, Port 2, and Port 3. Port 0 is nibble programmable as input, output, or address. Port 1 is byte configurable as input, output, or address/data. Port 2 is bit programmable as either inputs or outputs, with or without handshake and SPI. Port 3 can be programmed to provide timing, serial and parallel input/output, or comparator input/output.

All ports have push–pull CMOS outputs. In addition, the push–pull outputs of Port 2 can be turned OFF for open-drain operation.

Mode Registers

Each port has an associated Mode Register that determines the port's functions and allows dynamic change in port functions during program execution. Port and Mode Registers are mapped into the Standard Register File as displayed in Figure 28.

Register	HEX	Identifier
Port 0–1 Mode	F8h	P01M
Port 3 Mode	F7h	P3M
Port 2 Mode	F6h	P2M
Port 3	03h	P3
Port 2	02h	P2
Port 1	01h	P1
Port 0	00h	P0

Figure 28. I/O Ports and Mode Registers

Because of their close association, Port and Mode registers are treated like any other general-purpose registers. There are no special instructions for port manipulation. Any instruction which addresses a register can address the ports. Data can be directly accessed in the Port Register, with no extra moves.

Input and Output Registers

Each bit of Ports 0, 1, and 2, has an input register, an output register, associated buffer, and control logic. Because there are separate input and output registers associated with each port, writing to bits defined as inputs stores the data in the output register. This data cannot be read as long as the bits are defined as inputs. However, if the bits are reconfigured as outputs, the data stored in the output register is reflected on the output pins and can then be read. This mechanism allows you to initialize the outputs prior to driving their loads (see Figure 29 on page 45).

Because port inputs are asynchronous to the Z8® CPU internal clock, a READ operation could occur during an input transition. In this case, the logic level might be uncertain (between a logic 1 and 0). To eliminate this meta-stable condition, the Z8 CPU latches the input data two clock periods prior to the execution of the current instruction. The input register uses these two clock periods to stabilize to a legitimate logic level before the instruction reads the data.

> **Note:** *The following sections describe the generic function of the Z8 CPU ports. Any additional features of the ports such as SPI, C/T, and Stop Mode Recovery are described in the respective sections.*

Port 0

This section deals with only the I/O operation of Port 0. Figure 29 on page 45 displays a block diagram of Port 0. This diagram also applies to Ports 1 and 2.
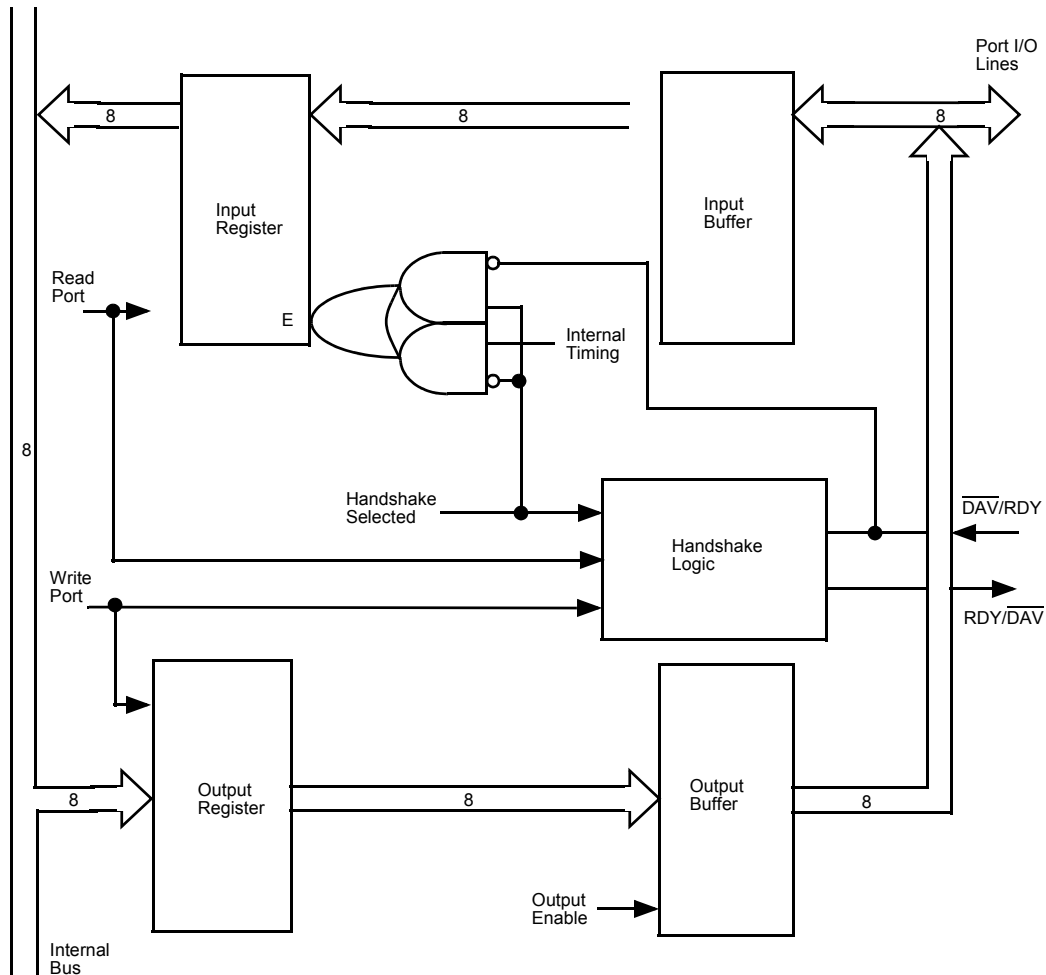
Figure 29. Ports 0, 1, 2 Generic Block Diagram

General I/O Mode

Port 0 can be an 8-bit, bidirectional, CMOS or TTL compatible I/O port. These eight I/O lines can be configured under software control as a nibble I/O port (P03–P00 input/output and P07–P04 input/output), or as an address port for interfacing external memory. The input buffers can be Schmitt-Triggered, level shifted, or a single-trip point buffer and can be nibble programmed. Either nibble output can be globally programmed as push–pull or open-drain. Low EMI output buffers in some cases can be globally programmed by the software as an OTP program option or as a ROM mask option. In such cases, the Z8® MCU features autolatches that are hardwired to the inputs. Refer to the specific Z8 MCU product specification for the exact input/output buffer features that are available (see Figure 30 on page 46 and Figure 31 on page 47).

Figure 30. Port 0 Configuration with Open-Drain Capability, Autolatch, and Schmitt-Trigger
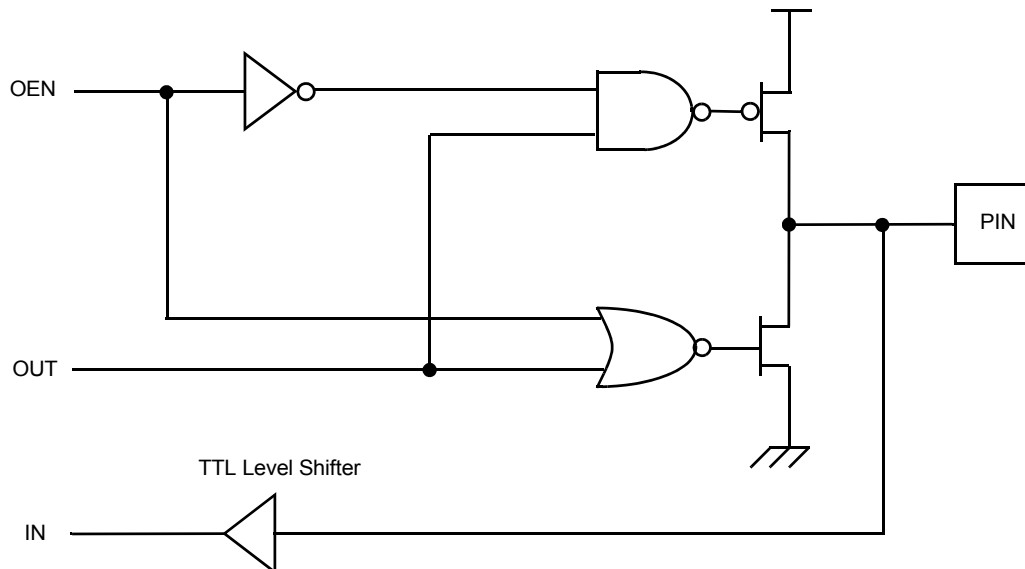
Figure 31. Port 0 Configuration with TTL Level Shifter

Read/Write Operations

In the nibble I/O Mode, Port 0 is accessed as general-purpose register P0 (`00h`) with ERF Bank set to 0. The port is written by specifying P0 as an instruction's destination register. Writing to the port causes data to be stored in the port's output register.

The port is read by specifying P0 as the source register of an instruction. When an output nibble is read, data on the external pins is returned. Under normal loading conditions this is equivalent to reading the output register. However, for Port 0 outputs defined as open–drain, the data returned is the value forced on the output by the external system. This may not be the same as the data in the output register. Reading a nibble defined as input also returns data on the external pins. However, input bits under handshake control return data latched into the input register via the input strobe.

The Port 0–1 Mode resistor bits D1–D0 and D7–D6 are used to configure Port 0 nibbles. The lower nibble (P00–P03) can be defined as inputs by setting bits D1 to 0 and D0 to 1, or as outputs by setting both D1 and D0 to 0. Likewise, the upper nibble (P04–P07) can be defined as inputs by setting bits D7 to 0 and D6 to 1, or as outputs by setting both D6 and D7 to 0 (see Figure 32 on page 49).

Handshake Operation

When used as an I/O port, Port 0 can be placed under handshake control by programming the Port 3 Mode register bit D2 to 1. In this configuration, handshake control lines are

DAV0 (P32) and RDY0 (P35) when Port 0 is an input port, or RDY0 (P32) and DAV0 (P35) when Port 0 is an output port (see Figure 33 on page 49).

Handshake direction is determined by the configuration (input or output) assigned to the Port 0 upper nibble:P04–P07. The lower nibble must have the same I/O configuration as the upper nibble to be under handshake control. Figure 30 on page 46 displays the Port 0 upper and lower nibbles and the associated handshake lines of Port 3.

Port 1

This section describes only the I/O operation. The port's external memory interface operation is discussed later in this manual. Figure 29 on page 45 displays a block diagram of Port 1.

General I/O Mode

Port 1 can be an 8-bit, bidirectional, CMOS or TTL compatible port with multiplexed address (A7–A0) and data (D7–D0) ports. These eight I/O lines can be byte programmed as inputs or outputs or can be configured under software control as an Address/Data port for interfacing to external memory. The input buffers can be Schmitt-Triggered, level-shifted, or a single-point buffer. In some cases, the output buffers can be globally programmed as either push–pull or open-drain. Low-EMI output buffers can be globally programmed by software, as an OTP program option, or as a ROM Mask Option. In some cases, the Z8® MCU can have autolatches hardwired to the inputs. Refer to specific product specifications for exact input/output buffer-type features available (see Figure 32 and Figure 33 on page 49).

Register F8h (P01M)
Port 0–1 Mode Register (P01M)
(Write-Only)

| D7 | D6 | | | | | D1 | D0 |

P04–P07 Mode
00 = Output
01 = Input
1X = A12–A15

P00–P03 Mode
00 = Output
01 = Input
1X = A8–A11
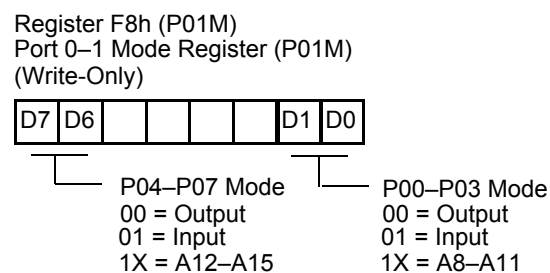
Figure 32. Port 0 I/O Operation

Register F7h
Port 3 Mode Register (P3M)
(Write-Only)

D2

0 P32 = Input
  P35 = Output
1 P32 = DAV0/RDY0
  P35 = RDY0/DAV0
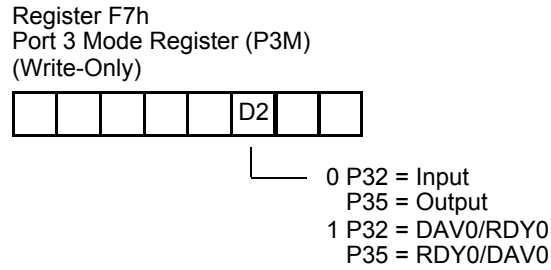
Figure 33. Port 0 Handshake Operation

OPEN-DRAIN

OEN

OUT

1.5 ←→ 2.3 V Hysteresis

IN

R ≈ 500 KΩ

Autolatch

PIN

Port 1
(I/O or AD7–AD0)
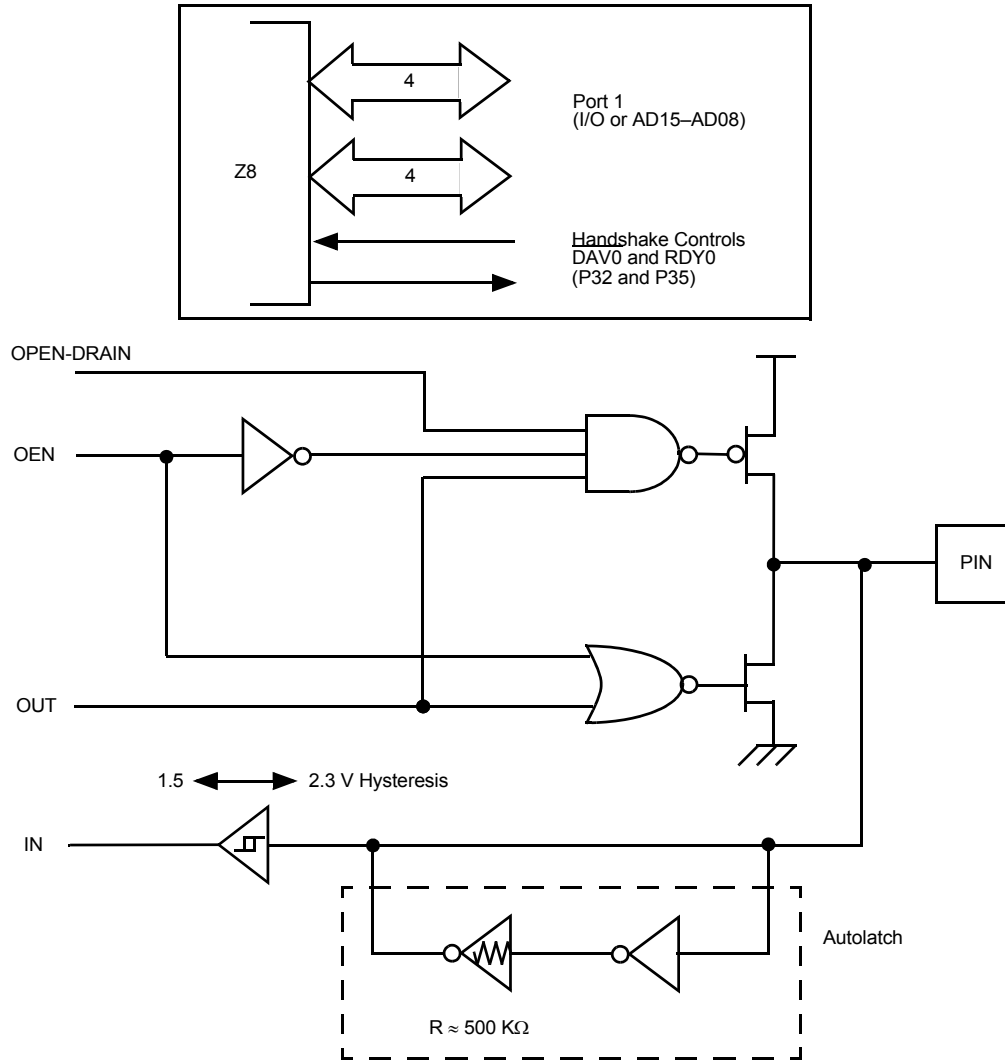
8

Handshake Controls
DAV1 and RDY1
(P33 and P34)

Z8
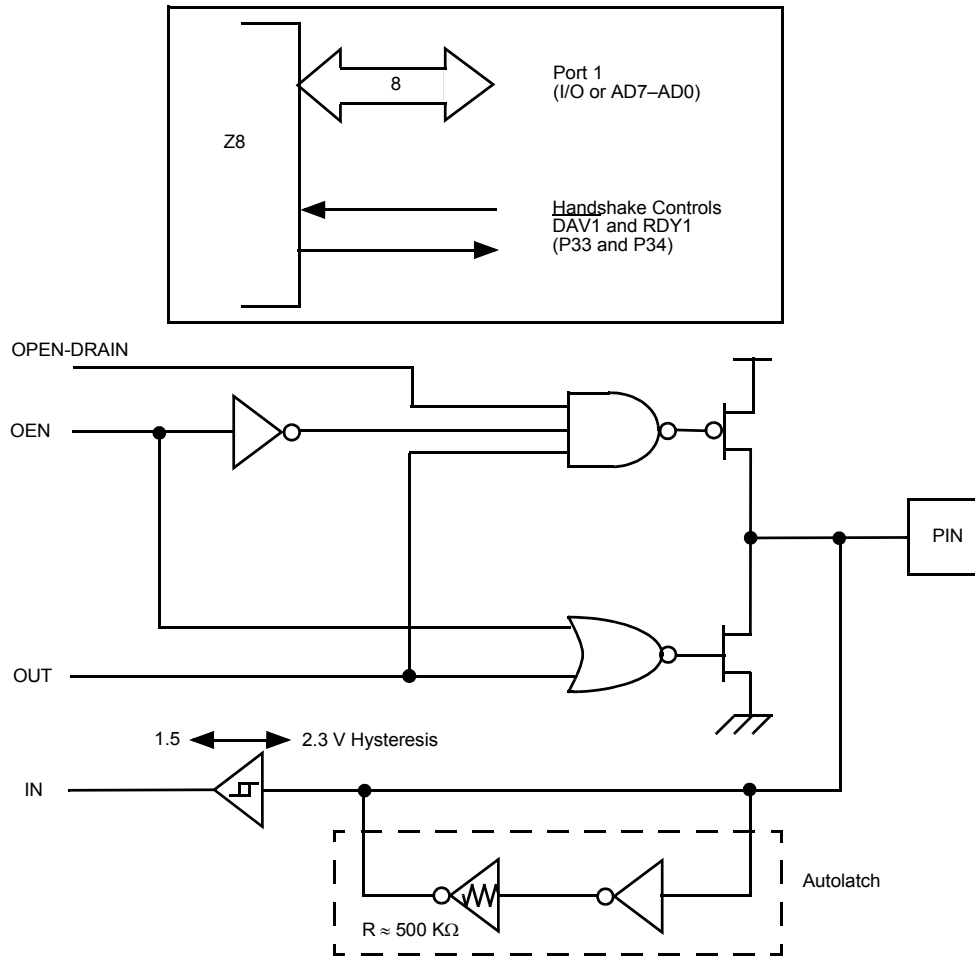
**Figure 34. Port 1 Configuration with Open-Drain Capability, Autolatch, and
Schmitt-Trigger**
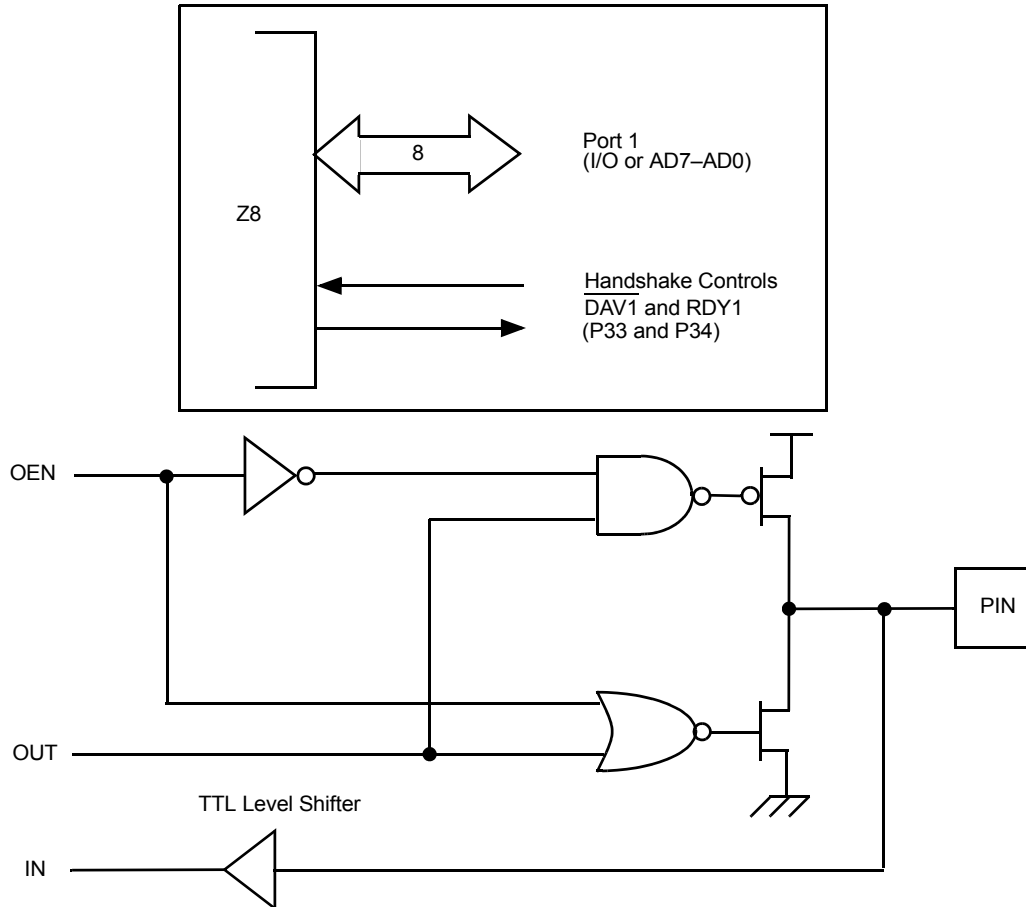
Figure 35. Port 1 Configuration with TTL Level Shifter

Read/Write Operations

In byte input or byte output mode, the port is accessed as General-Purpose Register P1 (01h). The port is written by specifying P1 as an instruction's destination register. Writing to the port causes data to be stored in the port's output register.

The port is read by specifying P1 as the source register of an instruction. When an output is read, data on the external pins is returned. Under normal loading conditions, this is equivalent to reading the output register. However, if Port 1 outputs are defined as open-drain, the data returned is the value forced on the output by the external system. This may not be the same as the data in the output register. When Port 1 is defined as an input, reading also returns data on the external pins. However, inputs under handshake control return data latched into the input register via the input strobe.

Using the Port 0–1 Mode Register, Port 1 is configured as an output port by setting bits D4 and D3 to 0, or as an input port by setting D4 to 0 and D3 to 1 (see Figure 36).

R248 P01M
Port 0–1 Mode Register
(F8, Write-Only)

| | | | D4 | D3 | | | |

P10–P13 Mode
00 = Byte Output
01 = Byte Output
10 = AD0-AD7
11 = High Impedance AD0–AD7,
$\overline{AS}$, $\overline{DS}$, $\overline{R}$/W,
A8–A11, A12–A15
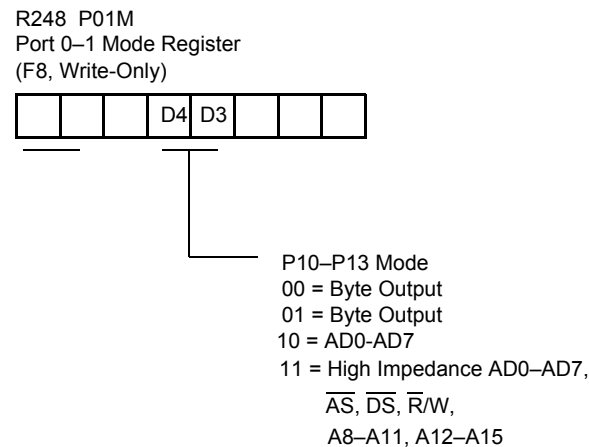
Figure 36. Port 1 I/O Operation

Handshake Operations

When used as an I/O port, Port 1 can be placed under handshake control by programming the Port 3 Mode register bits D4 and D3 both to 1. In this configuration, handshake control lines are DAV1 (P33) and RDY1 (P34) when Port 1 is an input port, or RDY1 (P33) and DAV1 (P34) when Port 1 is an output port. See Figure 37 through Figure 39 on page 53.

Handshake direction is determined by the configuration (input and output) assigned to Port 1. For example, if Port 1 is an output port then handshake is defined as output.

R247 P3M
Port 3 Mode Register
(F7, Write-Only)

| | | | D4 | D3 | | | |

00 P33 = Input   P34 = Output
01 P33 = Input   P34 = DM
10 P33 = Input   P34 = DM
11 P33 = DAV1/RDY1   P34 = RDY1/DAV1
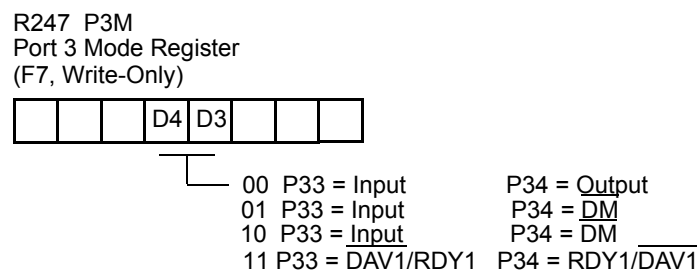
Figure 37. Handshake Operation

Port 2

Port 2 is a general-purpose port. Figure 29 on page 45 displays a block diagram of Port 2. Each of its lines can be independently programmed as input or output via the Port 2 Mode Register (F6h) as seen in Figure 38. A bit set to a 1 in P2M configures the corresponding bit in Port 2 as an input, while a bit set to 0 configures an output line.

Register F6h
Port 2 Mode Register (P2M)
(Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port 2 Mode
0 = Output
1 = Input
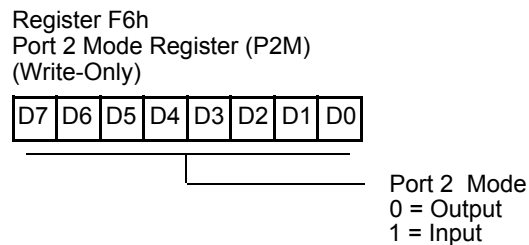
Figure 38. Port 2 I/O Mode Configuration

General Port I/O

Port 2 can be an 8-bit, bidirectional, CMOS- or TTL- compatible I/O port. These eight I/O lines can be configured under software control to be an input or output, independently. Input buffers can be Schmitt-Triggered, level-shifted, or a single trip point buffer and may contain autolatches. Bits programmed as outputs may be globally programmed as either push-pull or open-drain. Low-EMI output buffers can be globally programmed by the software, an OTP program option, or as a ROM mask option. In addition, when the SPI is featured and enabled, P20 functions as data-in (DI), and P27 functions as data-out (DO). Refer to specific product specifications for exact input/output buffer type features available. See Figure 39 on page 53 through Figure 41 on page 54.
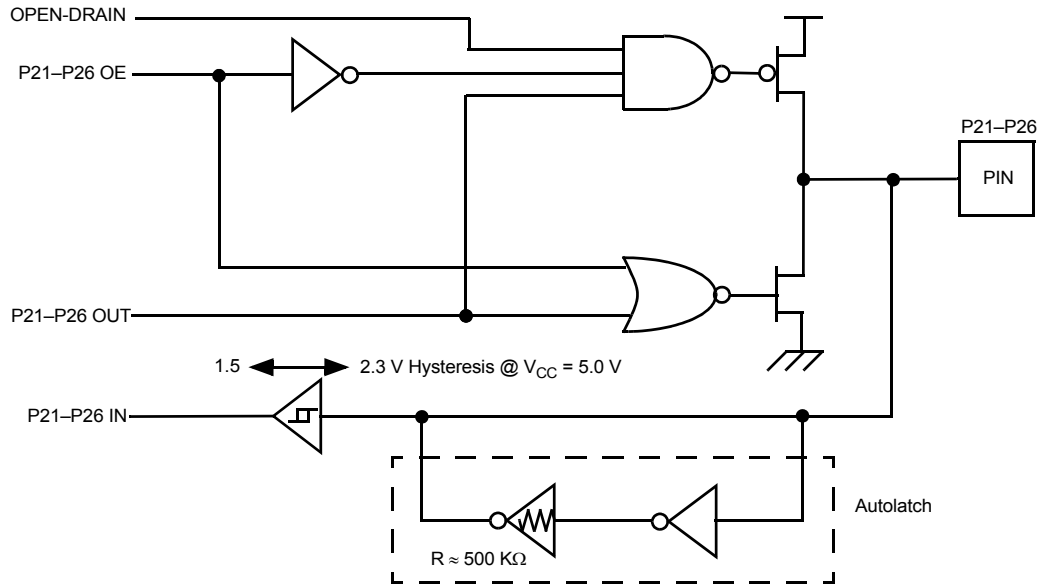
Figure 39. Port 2 Configuration with Open-Drain Capability, Autolatch, and Schmitt-Trigger
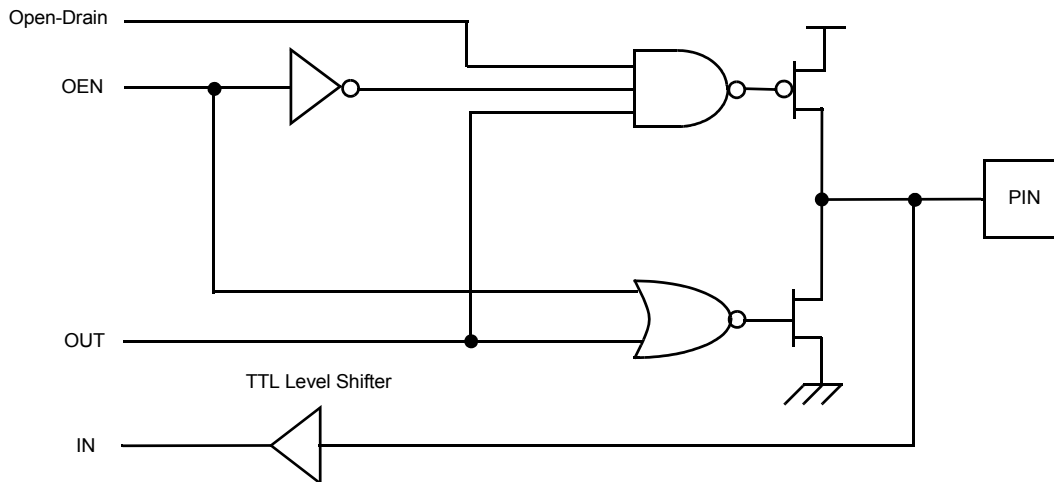
Figure 40. Port 2 Configuration with TTL Level Shifter
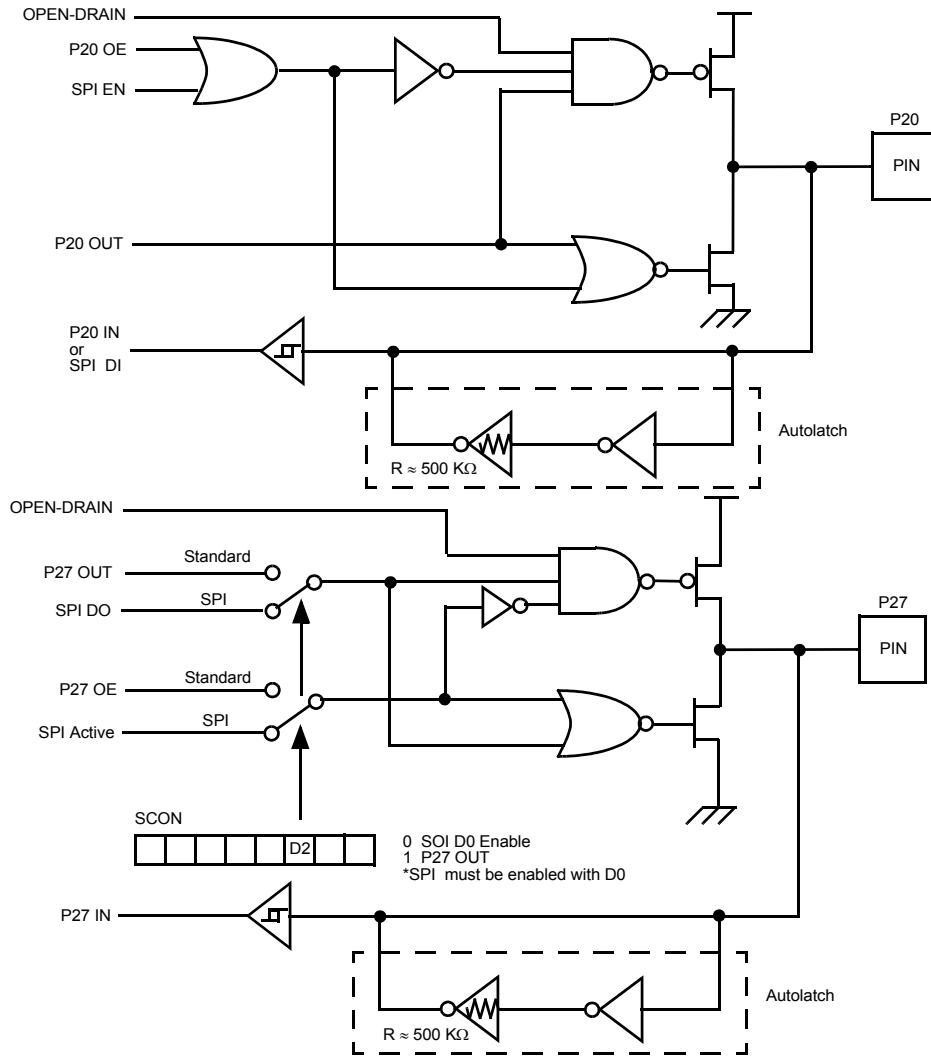
Figure 41. Port 2 Configuration with Open-Drain Capability, Autolatch, Schmitt-Trigger and SPI

Read/Write Operations

Port 2 is accessed as General-Purpose Register P2 (02h). Port 2 is written by specifying P2 as an instruction's destination register. Writing to Port 2 causes data to be stored in the output register of Port 2, and reflected externally on any bit configured as an output.

Port 2 is read by specifying P2 as the source register of an instruction. When an output bit is read, data on the external pin is returned. Under normal loading conditions, this is equivalent to reading the output register. However, if a bit of Port 2 is defined as an

open-drain output, the data returned is the value forced on the output pin by the external system. This may not be the same as the data in the output register. Reading input bits of Port 2 also returns data on the external pins. However, inputs under handshake control return data latched into the input register via the input strobe.

Handshake Operation

Port 2 can be placed under handshake control by programming bit 6 in the Port 3 Mode Register (see Figure 42). In this configuration, Port 3 lines P31 and P36 are used as the handshake control lines $\overline{\text{DAV2}}$ and RDY2 for input handshake, or RDY2 and $\overline{\text{DAV2}}$ for output handshake.

Handshake direction is determined by the configuration (input or output) assigned to bit 7 of Port 2. Only those bits with the same configuration as P27 are under handshake control. Figure 43 displays the bit lines of Port 2 and the associated handshake lines of Port 3.

Register F7h
Port 3 Mode Register
(Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port 2 Handshaking
0 P31 = Input (T_{IN})   P36 = Output (T_{OUT})
1 P31 = $\overline{\text{DAV2}}$/RDY2   P36 = RDY2/$\overline{\text{DAV2}}$
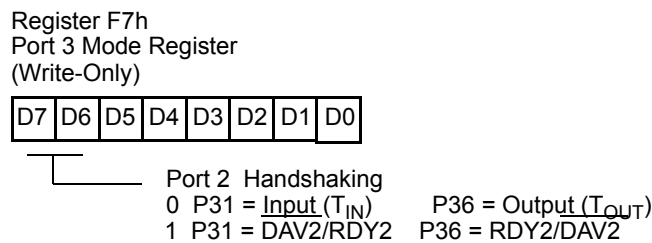
Figure 42. Port 2 Handshake Configuration

P20

Port 2 (I/O)

P27

Handshake Controls
$\overline{\text{DAV2}}$ and RDY2
(P31 and P36)
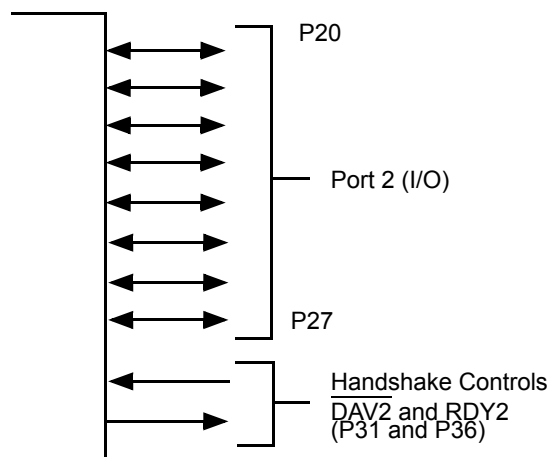
Figure 43. Port 2 Handshaking

Port 3

General Port I/O

Port 3 differs structurally from Ports 0, 1, and 2. Port 3 lines are fixed as four inputs (P33–P30) and four outputs (P37–P34) Port 3 does not have an input and output register for each bit. Instead, all of the input lines have one input register, and all of the output lines have an output register. Port 3 can be a CMOS- or TTL- compatible I/O port. Under software control, the lines can be configured as special control lines for handshake, comparator inputs, SPI control, external memory status, or I/O lines for the on-board serial and timer facilities. Figure 44 on page 57 displays the block diagram of Port 3.

The inputs can be Schmitt-Triggered, level-shifted, or single-trip point buffered. In some cases, the Z8® MCU may have autolatches hardwired on certain Port 3 inputs and Low-EMI capabilities on the outputs. Refer to specific product specifications for exact input/output buffer type features. Refer to the sections on counter/timers, Stop Mode Recovery, serial I/O, comparators, and interrupts for more information on the relationships of Port 3 to that feature.
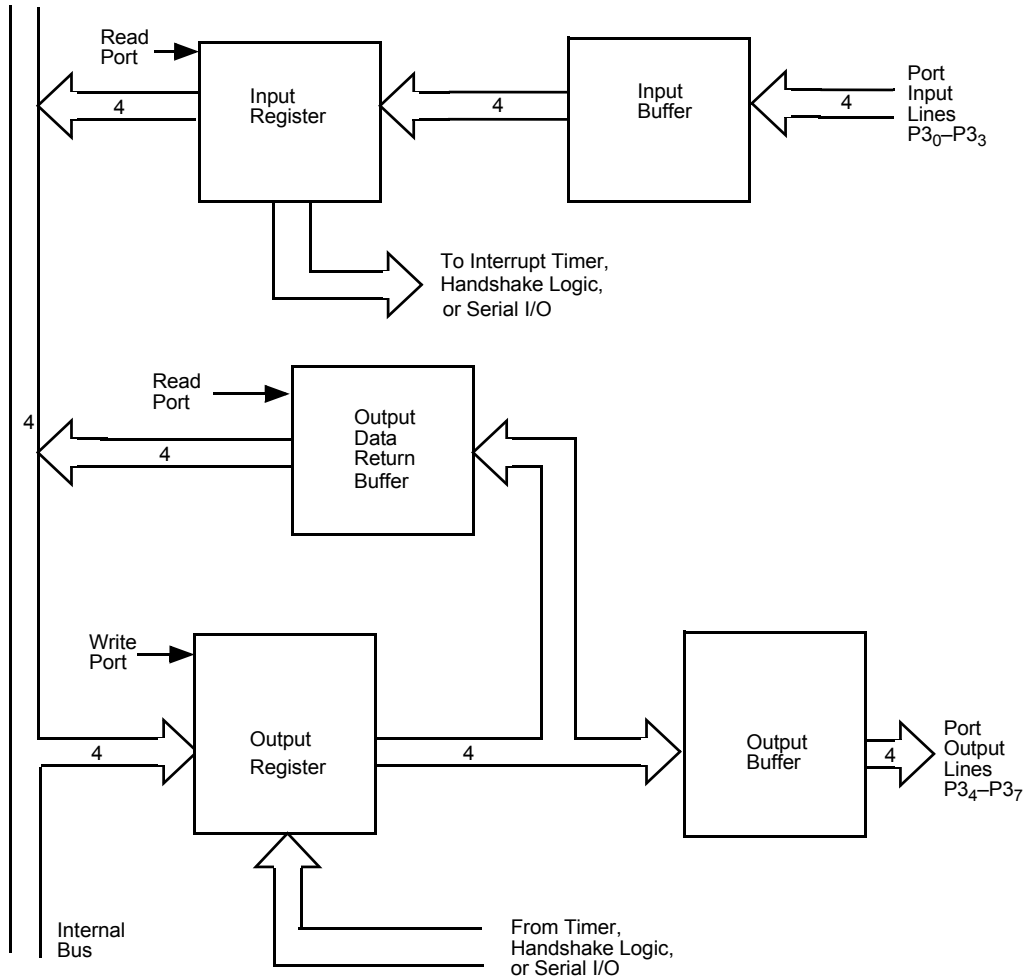
Figure 44. Port 3 Block Diagram
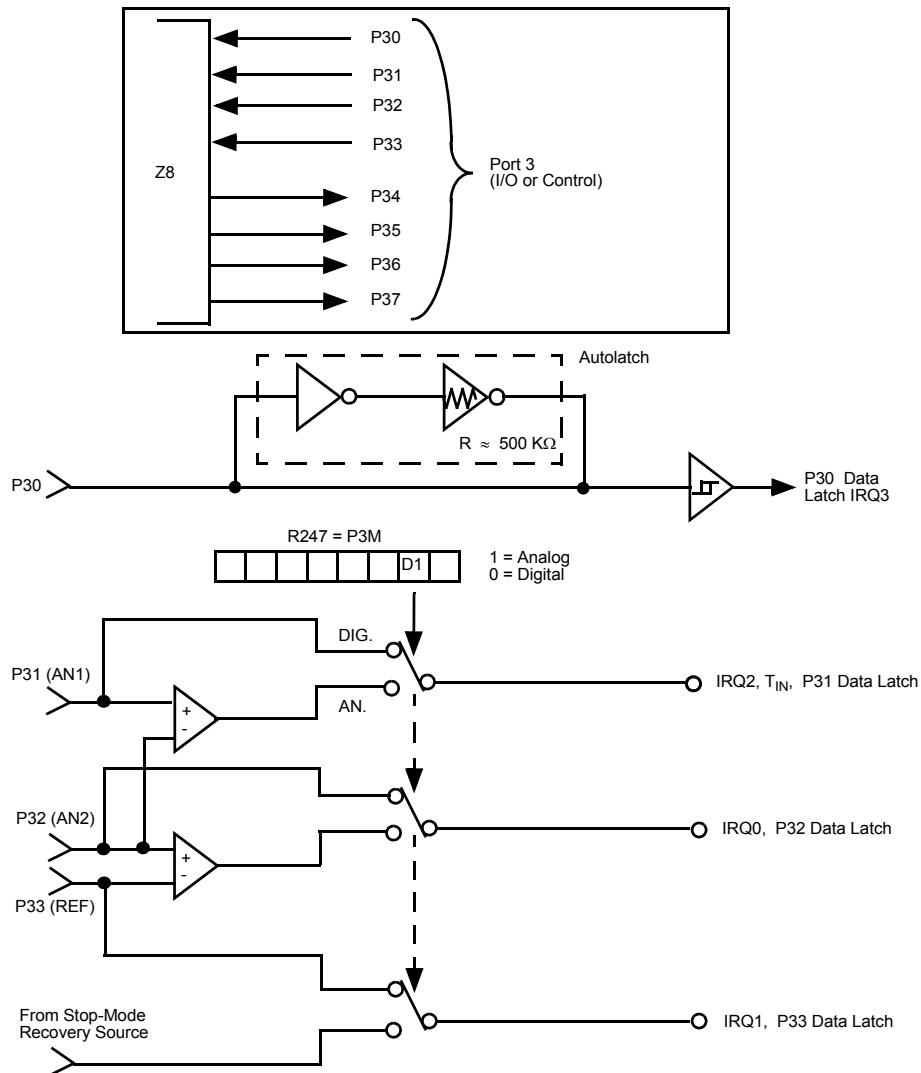
Figure 45. Port 3 Configuration with Comparator, Autolatch, and Schmitt-Trigger
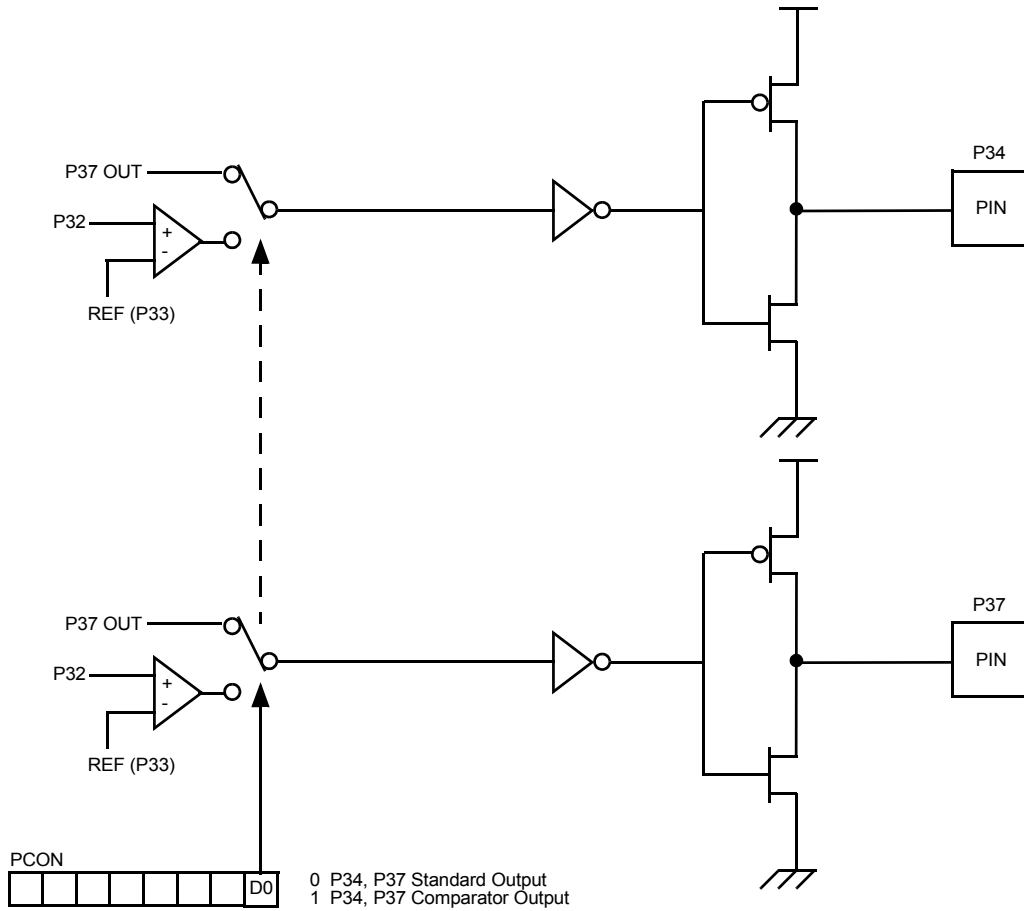
Figure 46. Port 3 Configuration with Comparator
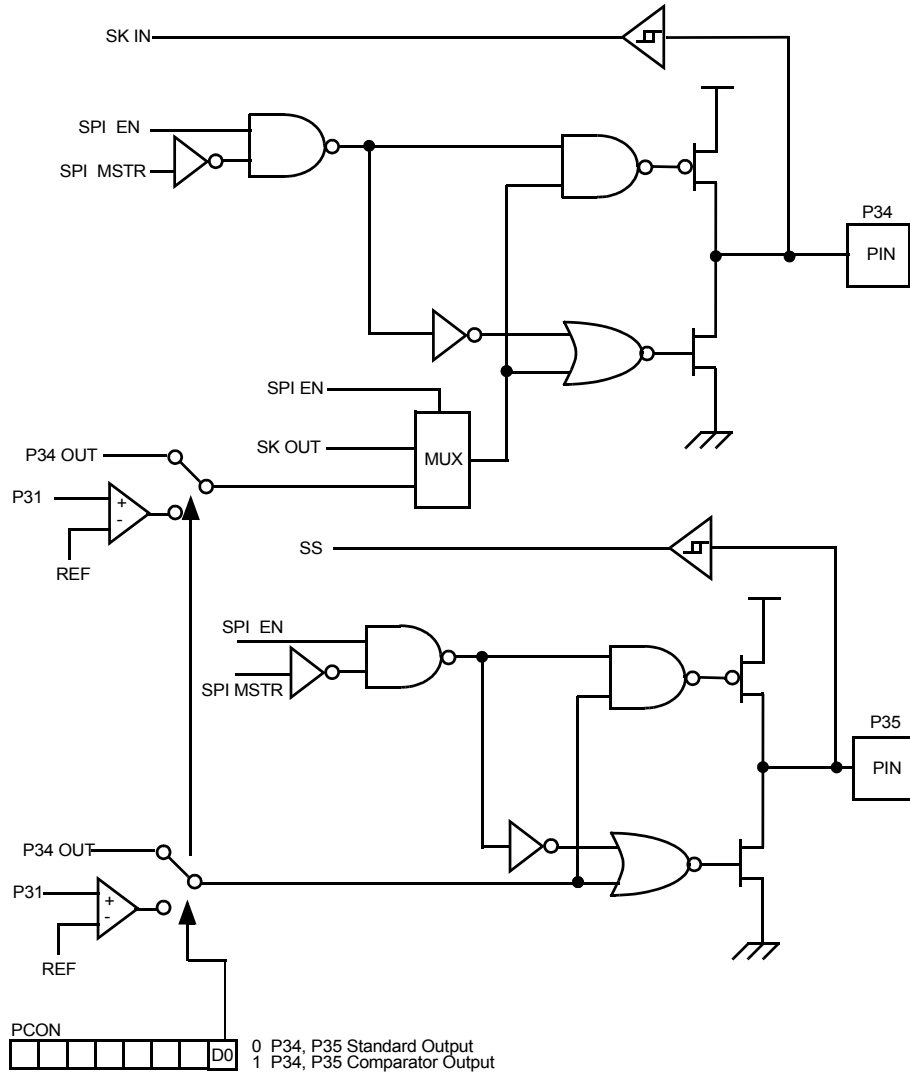
Figure 47. Port 3 Configuration with SPI and Comparator Outputs
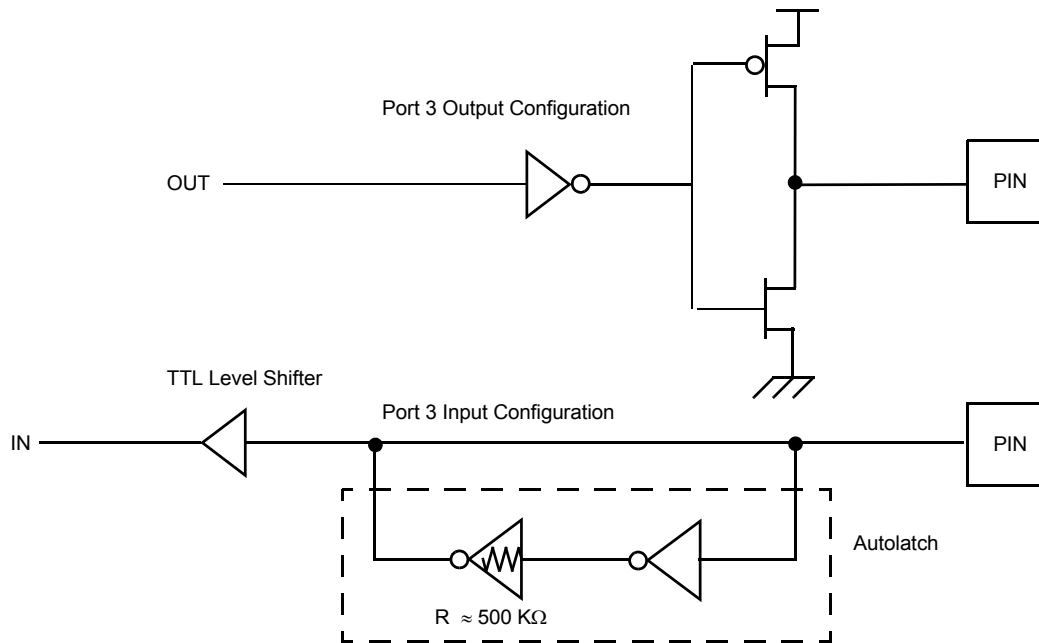
Figure 48. Port 3 Configuration with TTL Level Shifter and Autolatch

Read/Write Operations

Port 3 is accessed as a General-Purpose Register P3 (03h). Port 3 is written by specifying P3 as an instruction's destination register. However, Port 3 outputs cannot be written to if they are used for special functions. When writing to Port 3, data is stored in the output register.

Port 3 is read by specifying P3 as the source register of an instruction. When reading from Port 3, the data returned is both the data on the input pins and in the output register.

Special Functions

Special functions for Port 3 are defined by programming the Port 3 Mode Register. By writing 0s in bit 6 through bit 1, lines P37–P30 are configured as input/output pairs (see Figure 49 on page 62). Table 17 on page 62 lists available functions for Port 3. The special functions indicated in the figure are discussed in detail in their corresponding sections in this manual.

Port 3 input lines P33–P30 always function as interrupt requests regardless of the configuration specified in the Port 3 Mode Register.

Register F7h
Port 3 Mode Register
(Write-Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

0 Port 2 Open-Drain
1 Port 2 Push–Pull

0 P31, P32 Digital Mode
1 P31, P32 Analog Mode

0 P32 = Input   P35 = Output

00 P33 = Input   P34 = Output
01 P33 = Input   P34 = DM
10 P33 = Input   P34 = DM

0 P31 = Input   P36 = Output

0 P30 = Input   P37 = Output
1 P30 = Serial In   P37 = Serial Out

0 Party ON
1 Party OFF
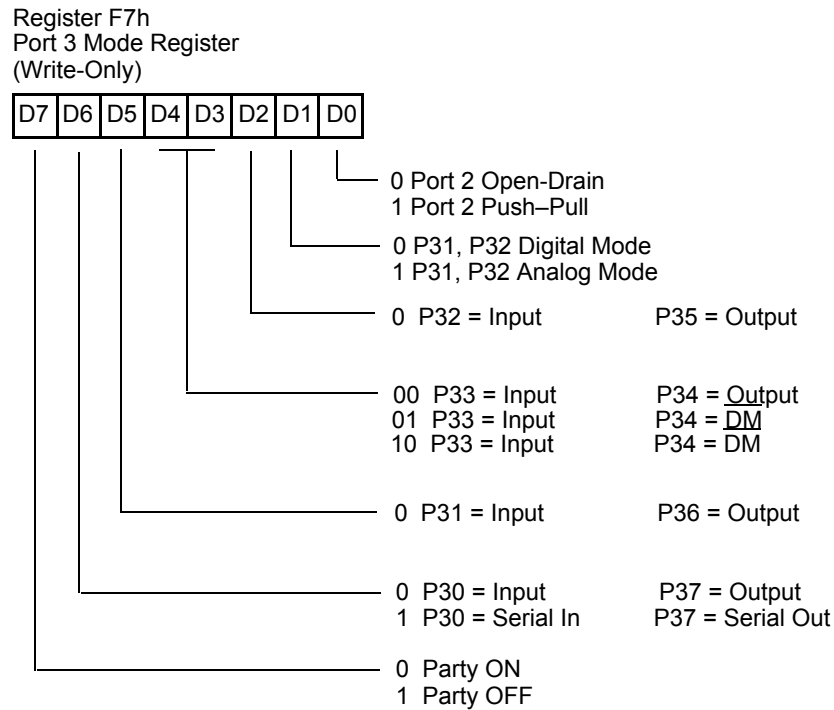
Figure 49. Port 3 Mode Register Configuration

Table 17. Port 3 Line Functions

Function	Line	Signal
Inputs	P30	Input
	P31	Input
	P32	Input
	P33	Input
Outputs	P34	Output
	P35	Output
	P36	Output
	P37	Output
Port 0 Handshake Input	P32	$\overline{\text{DAV0}}$/RDY0
Port 1 Handshake Input	P33	$\overline{\text{DAV1}}$/RDY1
Port 2 Handshake Input	P31	$\overline{\text{DAV2}}$/RDY2

Table 17. Port 3 Line Functions (Continued)

Function	Line	Signal
Port 0 Handshake Output	P35	RDY0/$\overline{\text{DAV0}}$
Port 1 Handshake Output	P34	RDY1/$\overline{\text{DAV1}}$
Port 2 Handshake Output	P36	RDY2/$\overline{\text{DAV2}}$
Analog Comparator Input	P31	AN1
	P32	AN2
	P33	REF
Analog Comparator Output	P34	AN1-OUT
	P35	AN2-OUT
	P37	AN2-OUT
Interrupt Requests	P30	$\overline{\text{IRQ3}}$
	P31	IRQ2
	P32	IRQ0
	P33	IRQ1
Serial Input (UART)	P30	DI
Serial Output (UART)	P37	DO
SPI Slave Select	P35	SS
SPI Clock	P34	SK
Counter/Timer	P31	T_{IN}
	P36	T_{OUT}
External Memory Status	P34	$\overline{\text{DM}}$

Port Handshake

When Ports 0, 1, and 2 are configured for handshake operation, a pair of lines from Port 3 are used for handshake controls. The handshake controls are interlocked to properly time asynchronous data transfers between Z8® and a peripheral. One control line ($\overline{\text{DAV}}$) functions as a strobe from the sender to indicate to the receiver that data is available. The second control line (RDY) acknowledges receipt of the sender's data, and indicates when the receiver is ready to accept another data transfer.

In input mode, data is latched into the Port's input register by the first $\overline{\text{DAV}}$ signal, and is protected from being overwritten if additional pulses occur on the $\overline{\text{DAV}}$ line. This overwrite protection is maintained until the port data is read. In output mode, data written to

the port is not protected and can be overwritten by the Z8 CPU during the handshake sequence. To avoid losing data, the software must not overwrite the port until the corresponding interrupt request indicates that the external device has latched the data.

The software can always read Port 3 output and input handshake lines, but cannot write to the output handshake line.

The following is the recommended setup sequence when configuring a Port for handshake operation for the first time after a reset:

- Load P01M or P2M to configure the port for input/output

- Load P3 to set the Output Handshake bit to a logic 1

- Load P3M to select HANDSHAKE mode for the port

Once a data transfer begins, the configuration of the handshake lines should not be changed until the handshake is completed.

Figure 50 and Figure 51 on page 65 display detailed operation for the handshake sequence.
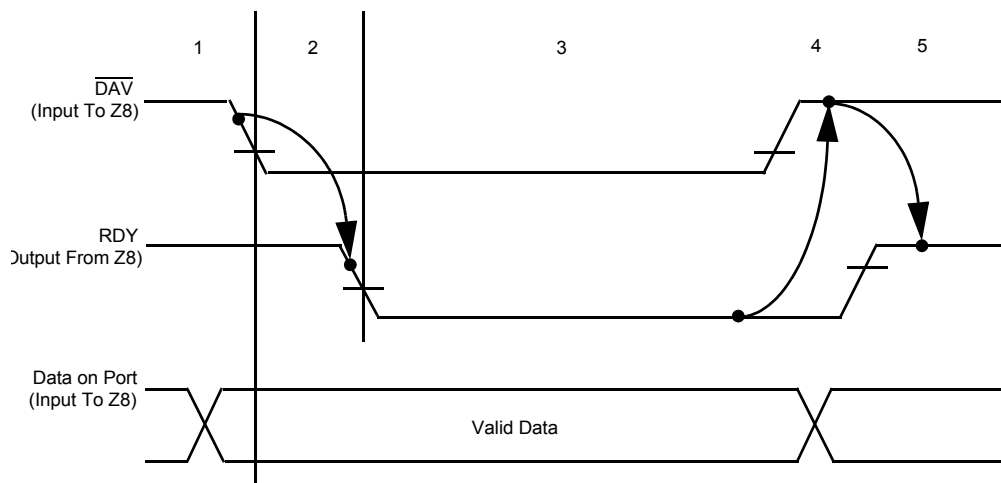
State 1.   Port 3 output is High, indicating that the I/O device is ready to accept data.

State 2.   The I/O device puts data on the port and then activates the $\overline{DAV}$ input. This causes the data to be latched into the port input register and generates an interrupt request.

State 3.   The Z8® CPU forces the Ready (RDY) output Low, signaling to the I/O device that the data has been latched.

State 4.   The I/O device returns the $\overline{DAV}$ line High in response to RDY going Low.

State 5.   The Z8 CPU RR software must respond to the interrupt request and read the contents of the port in order for the handshake sequence to be completed. The RDY line goes High if and only if the port has been read and $\overline{DAV}$ is High. This returns the interface to its initial state.

Figure 50. Z8 Input Handshake
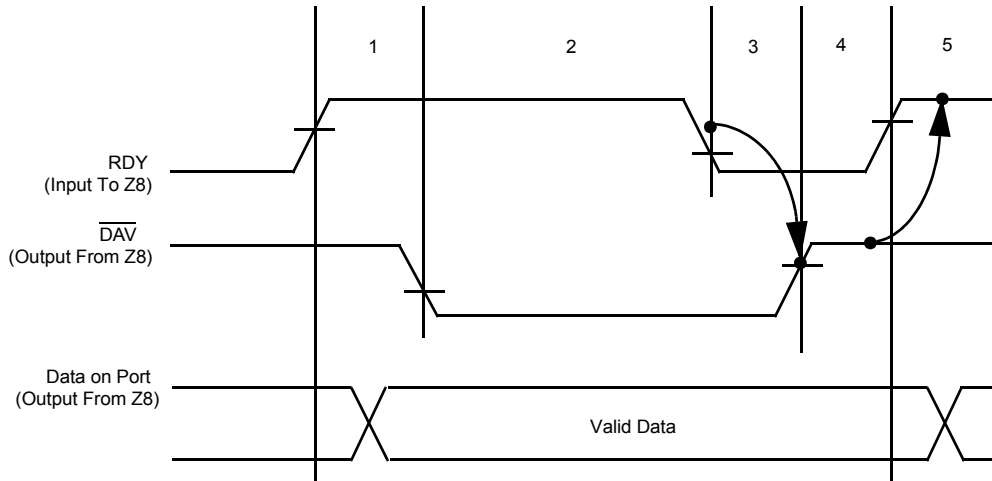
State 1.   RDY input is High indicating that the I/O device is ready to accept data.

State 2.   the Z8® CPU Writes to the port register to initiate a data transfer. Writing to the port outputs new data and forces DAV Low if and only if RDY is High.

State 3.   The I/O device forces RDY Low after latching the data. RDY Low causes an interrupt request to be generated the Z8 CPU can write new data responses to RDY going Low; however, the data is not output until State 5.

State 4.   The DAV output from the Z8 CPU is driven High in response to RDY going Low.

State 5.   The DAV goes High, the I/O device is free to raise RDY High thus returning the interface to its initial state.

Figure 51. Z8 Output Handshake

In applications requiring a strobed signal instead of the interlocked handshake, Z8® CPU can satisfy this requirement as follows:

- In the Strobed Input mode, data can be latched in the Port input register using the $\overline{DAV}$ input. The data transfer rate must allow enough time for the software to read the Port before strobing in the next character. The RDY output is ignored.

- In the Strobed Output Mode, the RDY input should be tied to the $\overline{DAV}$ output.

Figure 52 on page 66 and Figure 53 on page 66 display the strobed handshake connections.
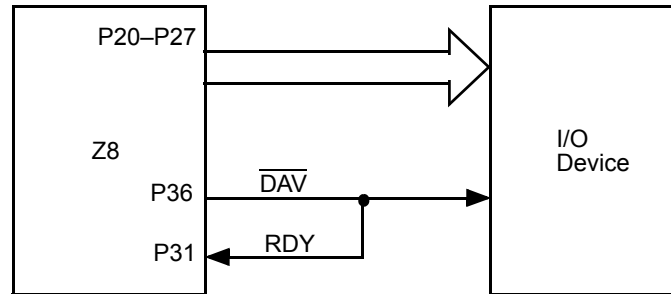
Figure 52. Output Strobed Handshake on Port 2
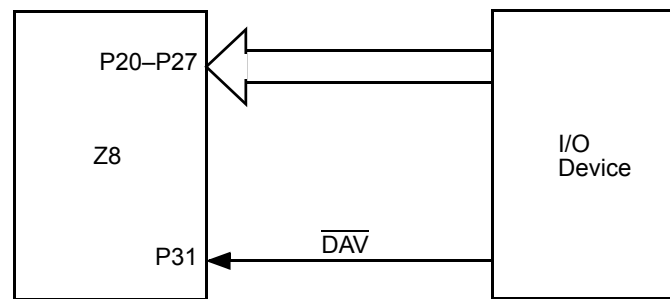
Figure 53. Input Strobed Handshake on Port 2

I/O Port Reset Conditions

Full Reset

After a hardware reset, WDT reset, or a POR, Port Mode Registers P01M, P2M, and P3M are set as displayed in Figure 54 on page 67 through Figure 56 on page 68. Port 2 is configured for input operation on all bits and is set for open-drain (see Figure 55 on page 67). If push-pull outputs are required for Port 2 outputs, remember to configure them using P3M. Note that a WDT time-out from Stop Mode Recovery does not do a full reset. Certain registers that are not reset after Stop Mode Recovery will not be reset.

For the condition of the Ports after Stop Mode Recovery, refer to the specific device product specifications. In some cases, Z8® MCU features the P01M, P2M, and P3M control register set back to the default condition after reset while others do not.

All special I/O functions of Port 3 are inactive, with P33–P30 set as inputs and P37–P34 set as outputs (see Figure 56 on page 68).

▶ **Note:** *Because the types and amounts of I/O vary greatly among the Z8® CPU family devices, it is recommended to review the selected device's product specifications for the register default state after reset.*

Register F8h
Port 0–1 Mode Register (P01M)
(Write-Only)

0	1	0	0	1	1	0	1

P00–P03 Mode
00 = Output
01 = Input
1X = A8–A11

Stack Selection
0 = External
1 = Internal

P10–P17 Mode
00 = Byte Output
01 = Byte Output
10 = AD0–AD7
11 = High Impedance AD0–AD7,
A8–A15, $\overline{AS}$, $\overline{DS}$, $\overline{R/W}$

External Memory Timing
Normal = 0
Extended = 1

P04–P07 Mode
Output = 00
Input = 01
A12–A15 = 1X

Figure 54. Port 0/1 Reset

Register F6h
Port 2 Mode Register (P2M)
(Write-Only)

1	1	1	1	1	1	1	1

Port 2 Mode
0 = Output
1 = Input

Figure 55. Port 2 Reset

Register F7h
Port 3 Mode Register (P3M)
(Write-Only)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0 Port 2 Open-Drain
1 Port 2 Push–Pull

0 P31, P32 Digital Mode
1 P31, P32 Analog Mode

0 P32 = <u>Input</u>   P35 = Outpu<u>t</u>
1 P32 = DAV0/RDY0   P35 = RDY0/DAV0

00 P33 = Input   P34 = <u>Output</u>
01 P33 = Input   P34 = <u>DM</u>
10 P33 = <u>Input</u>   P34 = DM
11 P33 = DAV1/RDY1 P34 = RDY1/DAV1

0 P31 = <u>Input</u>   P36 = Outpu<u>t</u>
1 P32 = DAV2/RDY2   P36 = RDY2/DAV2

0 P30 = Input   P37 = Output
1 P30 = Serial In   P37 = Serial Out

0 Parity OFF
1 Parity ON

Figure 56. Port 3 Mode Reset

Analog Comparators

Select Z8 devices include two independent on-chip analog comparators. See the device product specification for feature availability and use. Port 3, Pins P31, and P32 each has a comparator front end. The comparator reference voltage, pin P33, is common to both comparators. In analog mode, the P31, and P32 are the positive inputs to the comparators and P33 is the reference voltage supplied to both comparators. In digital mode, pin P33 can be used as a P33 register input or IRQ1 source. P34, P35, or P37 may output the comparator outputs by software-programming the PCON Register bit D0 to 1.

Comparator Description

Two on-board comparators process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming the Port 3 Mode Register (P3M bit 1). For interrupt functions during analog mode, P31 and P32 are programmable as rising, falling, or both edge triggered interrupts (IRQ register bits 6 and 7).

Note: *P33 cannot generate an external interrupt while in this mode. P33 can only generate interrupts in DIGITAL mode.*

Port 3 inputs must be in digital mode if Port 3 is a Stop Mode Recovery source. The analog comparator is disabled in STOP mode.

P31 can be used as T_{IN} in analog or digital modes, but it must be referenced to P33, when in analog mode.

Register F7h
Port 3 Mode Register (P3M)
(Write-Only)

0 = Digital Mode P31, P32, P33
1 = Analog Mode P31, P32, P33
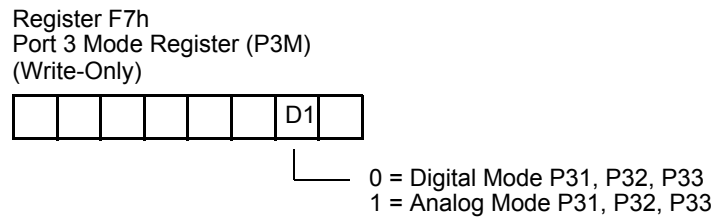
Figure 57. Port 3 Input Analog Selection

ERF Bank F
Register 00h
Port Configuration Register (PCON)
(Write-Only)

0 = P34, P35, or P37 Standard Outputs
1 = P34, P35, or P37 Comparator Outputs
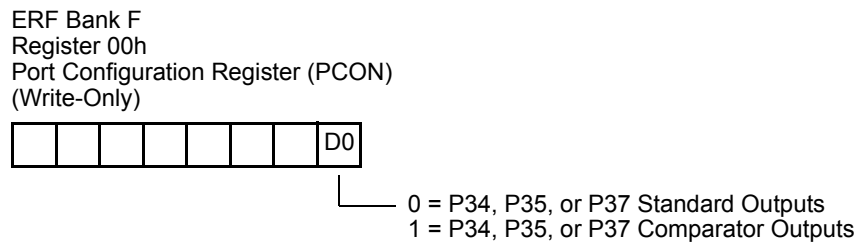
Figure 58. Port 3 Comparator Output Selection
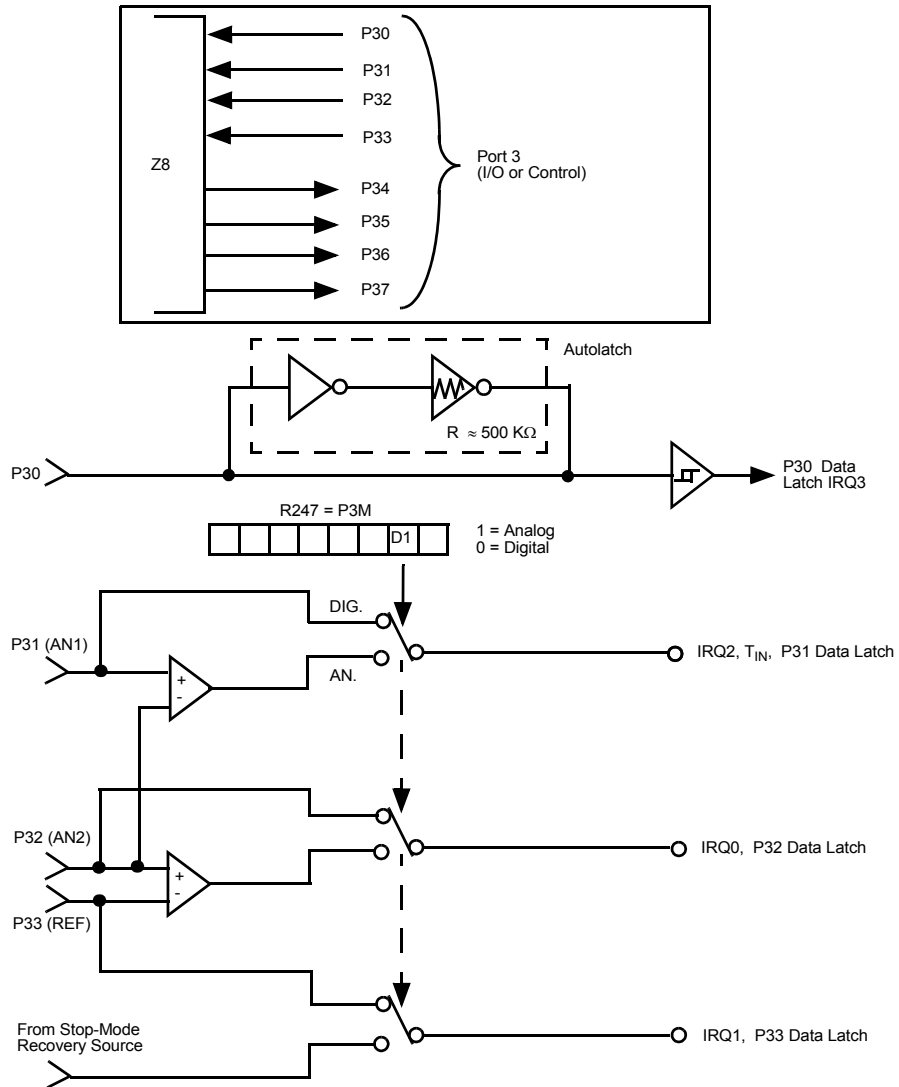
Figure 59. Port Configuration of Comparator Inputs on P31, P32, and P33
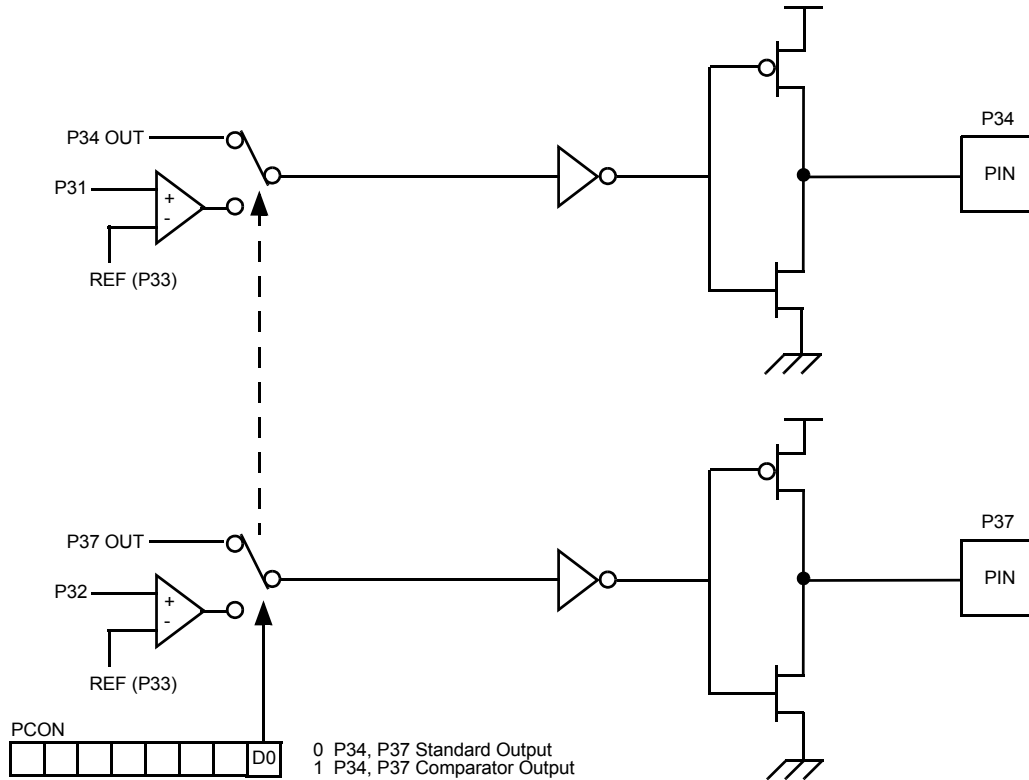
Figure 60. Port 3 Configuration

Comparator Programming

Example of enabling analog comparator mode.

```
LD P3M, #XXXX XX1Xb
```

X = any binary number

Example of enabling analog comparator output.

```
LD RP, #%0Fh          ;Sets register
                      pointer to
                      ;working register
                      group 0
                      ;and Expanded
                      Register
                      ;File Bank F.
```

```
LD R0, #XXXX XXX1b  ;Enables comparator
                    ;outputs using PCON
                    ;Register programming
```

Comparator Operation

After enabling the Analog Comparator mode, P33 becomes a common reference input for both comparators. The P33 (Ref) is hard wired to the reference inputs to both comparators and cannot be separated. P31 and P32 are always connected to the positive inputs to the comparators. P31 is the positive input to comparator AN1 while P32 is the positive input to comparator AN2. The outputs to comparators AN1 and AN2 are AN1-out and AN2-out, respectively.

The comparator output reflects the relationship between the positive input and the reference input.

Example

If the voltage on AN1 is higher than the voltage on Ref then AN1-out is at a high state. If voltage on AN2 is lower than the voltage on Ref then AN2-out is at a Low state. In this example, when the Port 3 register is read, Bits D1 = 1 and D2 = 0. If the comparator outputs are enabled to come out on P34 and P37, then P34 = 1 and P37 = 0.

> **Note:** *The previous data stored in P34 and P37 is not disturbed. Once the comparator outputs are de-selected the stored values in the P34 and P37 register bits are reflected on these pins again.*

Interrupts

P32 (AN2) will generate an interrupt based on the result of the comparison being low and the Interrupt Request Register (IRQ FAh) having bits D7 = 0 and D6 = 0. If IRQ D7 = 1 and D6 = 0 then both P31 and P32 would generate interrupts.

Comparator Definitions

V$_{ICR}$

The usable voltage range for both positive inputs and the reference input is called the common mode voltage range (V$_{ICR}$). The comparator is not guaranteed to work if the inputs are outside of the V$_{ICR}$ range.

V$_{OFFSET}$

The absolute value of the voltage between the positive input and the reference input required to make the comparator output voltage switch is the input offset voltage (V$_{OFFSET}$). If AN1 is 3.000 V and Ref is 3.001 V when the comparator output switches states then the V$_{offset}$ = 1 mV.

I$_{IO}$

For CMOS voltage comparator inputs, the input offset current (I$_{IO}$) is the leakage current of the CMOS input gate.

Run Mode

P33 is not available as an interrupt input during analog mode. P31 and P32 are valid interrupt inputs in conjunction with P33 (Ref) when in analog mode.

P31 can still be used as T$_{IN}$ when analog mode is selected. If comparator outputs are required to be outputted on the Port 3 outputs, refer to specific product specification for priority of mixing when other special features are sharing those same Port 3 pins.

Halt Mode

The analog comparators are functional during HALT mode if analog mode is enabled. P31 and P32, in conjunction with P33 (Ref) are able to generate interrupts. Only P33 cannot generate an interrupt because the P33 input goes directly to the Ref input of the comparators and is disconnected from the interrupt sensing circuits.

Stop Mode

The analog comparators are disabled during STOP mode so it does not use any current at that time. If P31, P32, or P33 are used as a source for Stop Mode Recovery, the Port 3 digital mode must be selected by setting bit D1 = 0 in the Port 3 Mode Register. Otherwise in STOP mode, the P31, P32, and P33 cannot be sensed. If analog mode is selected when entering STOP mode, it is still enabled after a valid SMR triggered reset.

Open-Drain Configuration

All Z8® MCUs can configure Port 2 to provide open-drain outputs by programming the Port 3 Mode Register (P3M) bit D0 = 0.

Register F7h
Port 3 Mode Register
(Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port 2 Configuration
1 = Pull-Ups Open-Drain
0 = Pull-Ups Active
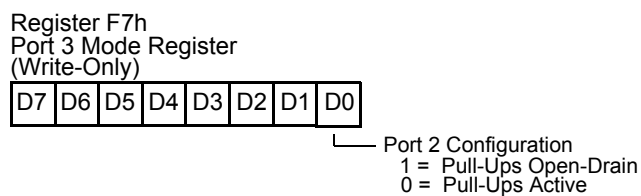
Figure 61. Port 2 Configuration

Other Z8 MCUs feature a Port Configuration Register (PCON) for which Port 0 and Port 1 can be configured to provide open-drain outputs. The PCON Register is located in ERF Bank F, Register 00h. See Figure 62 on page 74.

PCON (Fh) 00h

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Comparator Output Port 3
0 P34, P37 Standard Output*
1 P34, P37 Comparator Output

0 Port 1 Open Drain
1 Port 1 Push–pull Active *

0 Port 0 Open Drain
1 Port 0 Push–pull Active *

0 Port 0 Low EMI
1 Port 0 Standard *

0 Port 1 Low EMI
1 Port 1 Standard *

0 Port 2 Low EMI
1 Port 2 Standard *

0 Port 3 Low EMI
1 Port 3 Standard *

Low EMI Oscillator
0 Low EMI
* Default Setting After RESET   1 Standard *

Figure 62. Port Configuration Register (Write-Only)

Port 1 Open-Drain—Port 1, D1 can be configured as open-drain by resetting this bit (D1 = 0) or configured as push–pull active by setting this bit (D1 = 1). The default value is 1.

Port 0 Open Drain—Port 0, D2 can be configured as open-drain by resetting this bit (D2 = 0) or configured as push–pull active by setting this bit (D2 = 1). The default value is 1.

Low EMI Emission

Some Z8® MCUs can be programmed to operate in a Low EMI Emission Mode using the Port configuration register (PCON). The PCON register allows the oscillator and all I/O ports to be programmed in the Low-EMI Mode independently. Other Z8 MCUs may offer a ROM Mask or OTP programming option to configure the Z8 MCU ports and oscillator globally to a Low-EMI mode (where the XTAL frequency is set equal to the internal system clock frequency.

Use of the Low EMI feature results in:

- The output pre-drivers slew rate reduced to 10 ns (typical)

- Low EMI output drivers have resistance of 200Ω (typical)

- Low EMI Oscillator

- All output drivers are approximately 25 percent of the standard drive

- Internal SCLK ÷ TCLK = XTAL operation limited to a maximum of 4 MHz–250 ns cycle time, when Low EMI Oscillator is selected and system clock (SCLK = XTAL, SMR Reg. Bit D1 = 1)

For Z8® MCUs having the PCON register feature, the following bits control the Low EMI options.

Low EMI Port 0—Port 0, D3 can be configured as a Low EMI Port by resetting this bit (D3 = 0) or configured as a Standard Port by setting this bit (D3 = 1). The default value is 1.

Low EMI Port 1—Port 1, D4 can be configured as a Low EMI Port by resetting this bit (D4 = 0) or configured as a Standard Port by setting this bit (D4 = 1). The default value is 1.

Low EMI Port 2—Port 2, D5 can be configured as a Low EMI Port by resetting this bit (D5 = 0) or configured as a Standard Port by setting this bit (D5 = 1). The default value is 1.

Low EMI Port 3—Port 3, D6 can be configured as a Low EMI Port by resetting this bit (D6 = 0) or configured as a Standard Port by setting this bit (D6 = 1). The default value is 1.

Low EMI OSC—This D7 bit of the PCON Register controls the Low EMI oscillator. A 1 in this location configures the oscillator with standard drive, while a 0 configures the oscillator with low noise drive. The Low-EMI mode reduces the drive of the oscillator (OSC). The default value is 1. XTAL ÷ 2 mode is not affected by this bit.

Note: *The maximum external clock frequency is 4 MHz when running in the Low EMI oscillator mode.*

Refer to the selected device product specification for availability of the Low EMI feature and programming options.

Input Protection

All CMOS ROM Z8 MCUs have I/O pins with diode input protection. There is a diode from the I/O pad to V_{CC} and to V_{SS}. See Figure 63 on page 76.
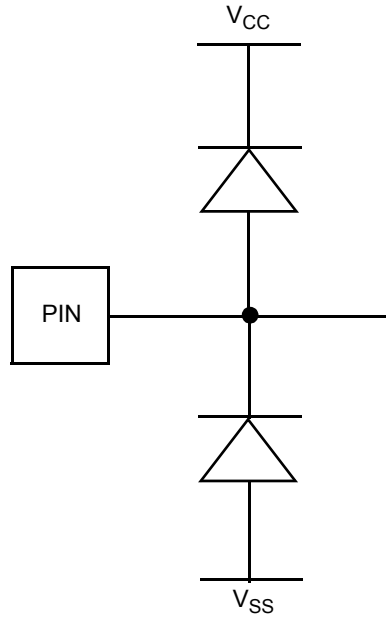
Figure 63. Diode Input Protection

On CMOS OTP EPROM Z8® MCUs, the Port 3 inputs P31, P32, P33, and the XTAL 1 pin have only the input protection diode from pad to V_{SS}. See Figure 64.
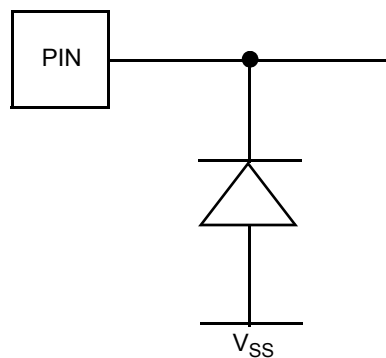
Figure 64. OTP Diode Input Protection

The high-side input protection diodes were removed on these pins to allow the application of +12.5 V during the various OTP programming modes.

For better noise immunity in applications that are exposed to system EMI, a clamping diode to V_{CC} from these pins may be required to prevent entering the OTP programming mode or to prevent high voltage from damaging these pins.

Z8® CMOS Autolatches

I/O port bits that are configurable as inputs are protected against open circuit conditions using autolatches. An autolatch is a circuit which, in the event of an open circuit condition, latches the input at a valid CMOS level. This inhibits the tendency of the input transistors to self-bias in the forward active region, thus drawing excessive supply current. A simplified schematic of the CMOS Z8 I/O circuit is displayed in Figure 65.
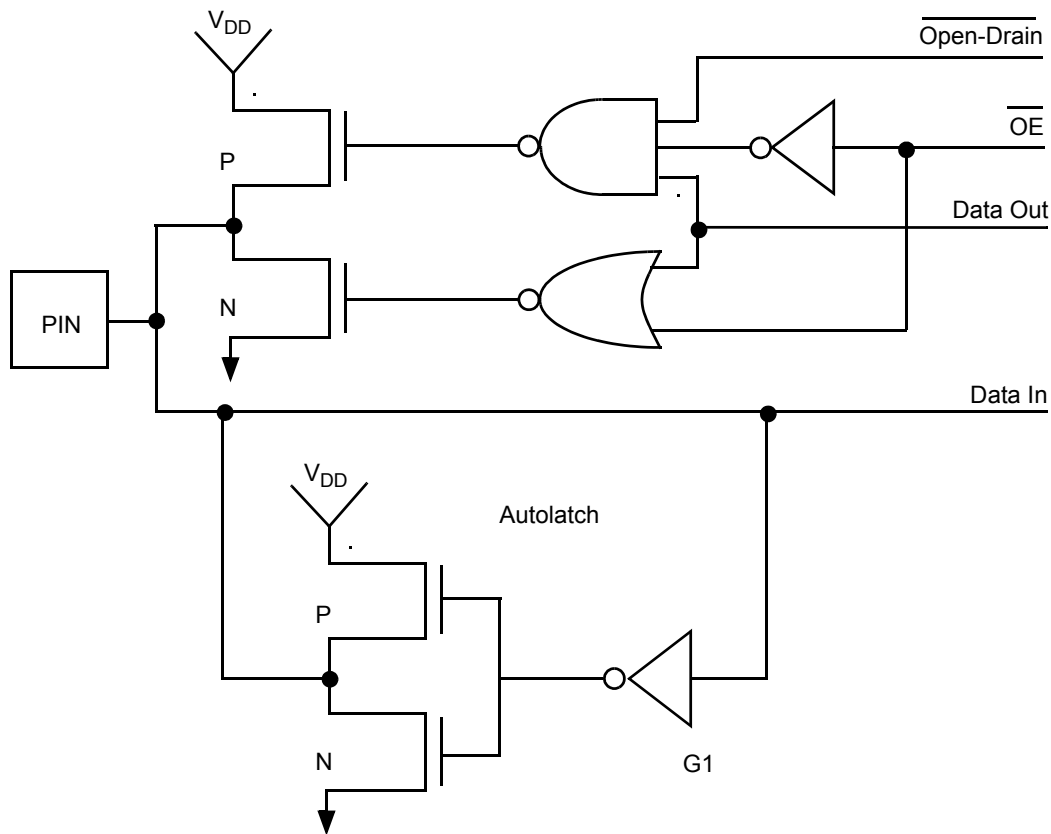
Figure 65. Simplified CMOS Z8 I/O Circuit

The operation of the autolatch circuit is straight-forward. Assume the input pad is latched at +5V (logic 1). The inverter G1 inverts the bit, turning the P-channel FET ON and the N-channel FET OFF. The output of the circuit is effectively shorted to V_{DD}, returning +5V to the input. If the pad is then disconnected from the +5V source, the autolatch hold the input at the previous state. If the device is powered up with the input floating, the state of the autolatch is at either supply, but which state is unpredictable.

There are four operating conditions which activate the autolatches. The first, which occurs when the input pin is physically disconnected from any source, is the most obvious. The

second occurs when the input is connected to the output of a device with tri-state capability.

The autolatch also activates when the input voltage at the pin is not within 200 microvolts or so of either supply rail. In this case, the circuit draws current, which is not significant compared to the I_{CC} operating current of the device, but increases I_{CC2} STOP mode current of the device dramatically.

The fourth condition occurs when the I/O bit is configured as an output. As displayed in Figure 65 on page 77, there are two ways of tri-stating the port pin. The first is by configuring the port as an input, which disables the $\overline{OE}$ signal turning both transistors OFF. The second can be achieved in output mode by writing a 1 to the output port, then activating the open-drain mode. Both transistors are again OFF, and the port bit is in a high impedance state. The autolatches then pull the input section toward V_{DD}.

Autolatch Model

The autolatch's equivalent circuit is displayed in Figure 66. When the input is high, the circuit consists of a resistance Rp from V_{DD} (the P-channel transistor in its ON state) and a much greater resistance Rh to G_{ND}. Current I_{AO} flows from V_{DD} to the output. When the input is low, the circuit may be modeled as a resistance Rp from G_{ND} (the N-channel transistor in the ON state) and a much greater resistance Rh to V_{DD}. Current I_{AO} now flows from the input to ground. The autolatch is characterized with respect to I_{AO}, so the equivalent resistance Rp is calculated according to $R_P = (V_{DD} - V_{IN})/I_{AO}$. The worst case equivalent resistance Rp (min) may be calculated at the worst case input voltage, $V_I = V_{IH}$ (min).
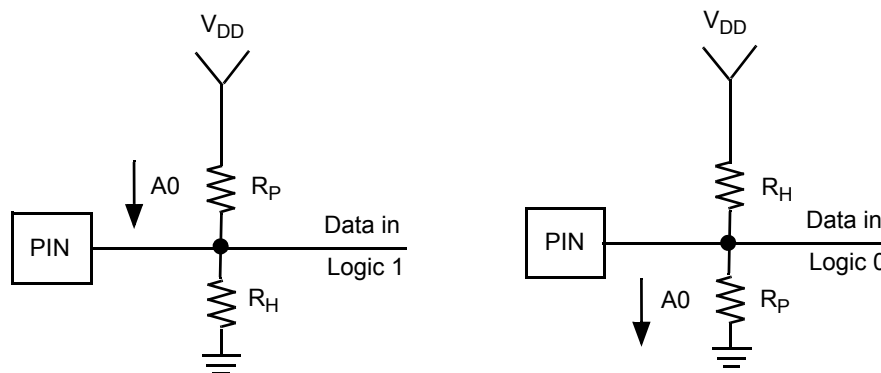
Figure 66. Autolatch Equivalent Circuit

Design Considerations

For circuits in which the autolatch is active, considerations should be given to the loading constraints of the autolatches. For example, with weak values of V_{IN}, close to Vih (min) or

Vil (max), pullup or pull-down resistances must be calculated using Ref = R/Rp. For best case STOP mode operation, the inputs should be within 200 mV of the supply rails.

In output mode, if a port bit is forced into a tri-state condition, the autolatches force the pad to V_{DD}. If there is an external pulldown resistor on the pin, the voltage at the pin may not switch to GND due to the autolatch. As displayed in Figure 67, the equivalent resistance of the autolatch and the external pulldown forms a voltage divider, and if the external resistor is large, the voltage developed across it will exceed V_{IL}(max). For worst case.

V_{IL}(max > V_{DD} [R_{EXT} ÷ (R_{EXT} + R_P)]
R_{EXT}(max) = [(V_{IL}(max) ÷ V_{DD}) R_P] ÷ [1—(V_{IL}(max) ÷ V_{DD})]

For V_{DD} = 5.0V and I_{AO} = 5µA, V_{IH}(max) = 0.8V:
R_{EXT}(max) = (0.16 ÷ 1M) ÷ (1—0.16) = 190K¾.

Rp increases rapidly with V_{DD}, so increased V_{DD} will relax the requirement on Rext.

In summary, the CMOS Z8 autolatch inhibits excessive current drain in Z8 devices by latching an open input to either V_{DD} or GND. The effect of the autolatch on the I/O characteristics of the device may be modeled by a current I_{AO} and a resistor Rp, whose value is V_{DD}/I_{AO}.
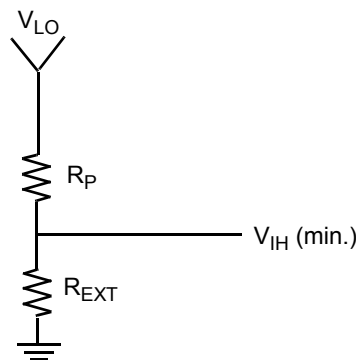
Figure 67. Effect of Pulldown Resistors on Autolatches

Counters and Timers

Z8® CPU provides up to two 8-bit counter/timers, T0 and T1, each driven by its own 6-bit prescaler, PRE0 and PRE1 (see Figure 68). Both counter/timers are independent of the processor instruction sequence, that relieves software from time-critical operations such as interval timing or event counting. Some MCUs offer clock scaling using the SMR. Refer to the device product specification for clock available options.

Each counter/timer operates in either Single-Pass or Continuous mode. At the end-of-count, counting either stops or the initial value is reloaded and counting continues. Under software control, new values are loaded immediately or when the end-of-count is reached. Software also controls the counting mode, how a counter/timer is started or stopped, and its use of I/O lines. Both the counter and prescaler registers can be altered while the counter/timer is running.
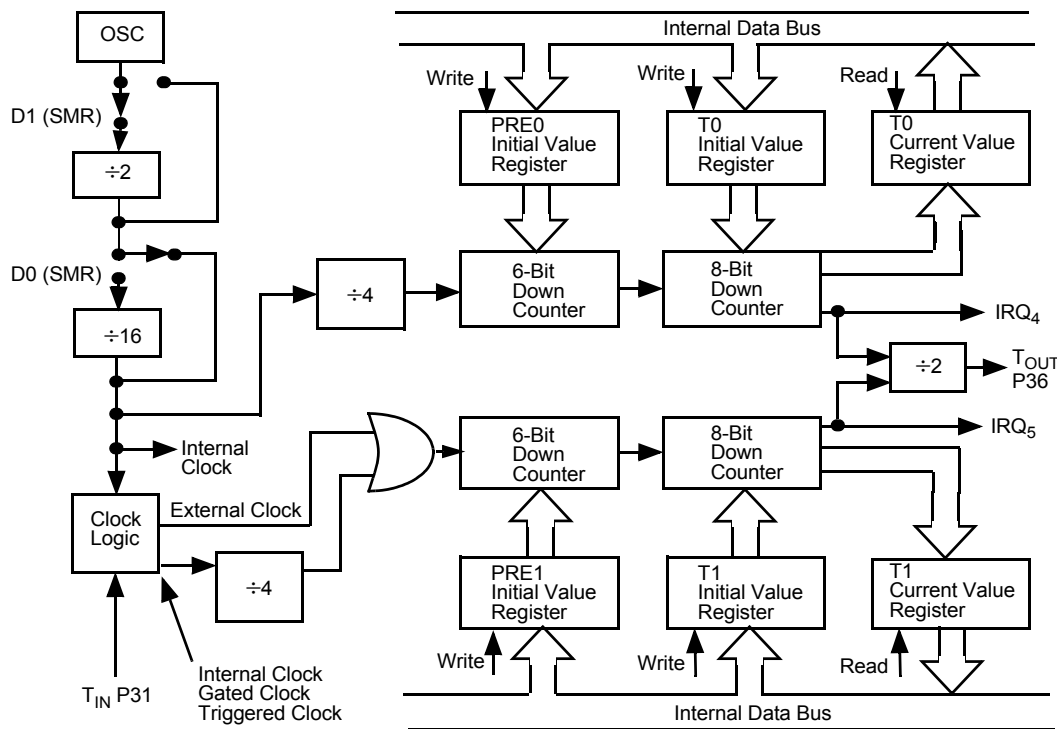
Figure 68. Counter/Timer Block Diagram

Counter/timers 0 and 1 are driven by a timer clock generated by dividing the internal clock by four. The divide-by-four stage, the 6-bit prescaler, and the 8-bit counter/timer form a synchronous 16-bit divide chain. Counter/timer 1 can also be driven by an external input

(T_{IN}) using P31. Port 3 line P36 can serve as a timer output (T_{OUT}) through which T0, T1, or the internal clock can be output. The timer output toggles at the end-of-count.

The counter/timer, prescaler, and associated mode registers are mapped into the register file as displayed in Figure 69. This allows the software to treat the counter/timers as general-purpose registers, and eliminates the requirement for special instructions.

Prescalers and Counter/Timers

The prescalers, PRE0 (F5h) and PRE1 (F3h), each consists of an 8-bit register and a 6-bit down-counter as displayed in Figure 68 on page 80. The prescaler registers are write-only registers. Reading the prescalers returns the value FFh. Figure 70 on page 82 and Figure 71 on page 82 displays the prescaler registers.

The six most significant bits (D2–D7) of PRE0 or PRE1 hold the prescalers count modulo, a value from 1 to 64 decimal. The prescaler registers also contain control bits that specify T0 and T1 counting modes. These bits also indicate whether the clock source for T_1 is internal or external. These control bits will be discussed in detail throughout this chapter.

The counter/timer registers, T0 (F4h) and T1 (F2h), each consists of an 8-bit down-counter, a write-only register that holds the initial count value, and a read-only register that holds the current count value (see Figure 68 on page 80). The initial value can range from 1 to 256 decimal (01h,02h,..,00h). Figure 72 on page 83 displays the counter/timer registers.

DEC	.	HEX Identifiers
247	Port 3 Mode	F7
245	T0 Prescaler	F5
244	Timer/Counter0	F4
243	T1 Prescaler	F3
242	Time/Counter1	F2
241	Timer Mode	F1

Figure 69. Counter/Timer Register Map

R245 PRE0
Prescaler 0 Register
(%F5; Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

Count Mode
$0 = T_0$ Single Pass
$1 = T_0$ Modulo-n

Reserved (Must be 0)

Prescaler Modulo
(Range: 1-64 Decimal
 01-00 HEX)
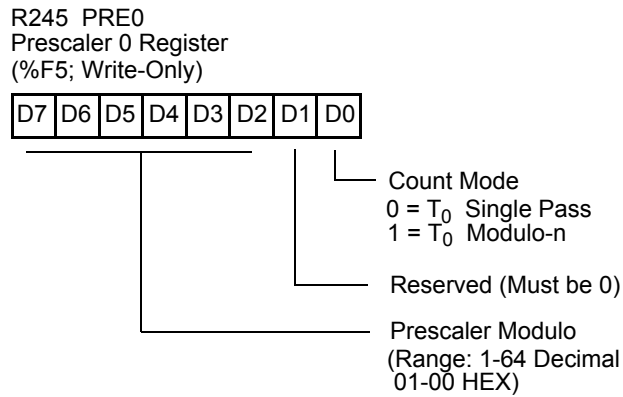
Figure 70. Prescaler 0 Register

R243 PRE1
Prescaler 1 Register
(%F3; Write-Only)

U	U	U	U	U	U	0	0

Count Mode
$0 = T_1$ Single Pass
$1 = T_1$ Modulo-n

Clock Source
$1 = T_1$ Internal
$0 = T_1$ External (T_{IN})

Prescaler Modulo
(Range: 1-64 Decimal
 01-00 HEX)
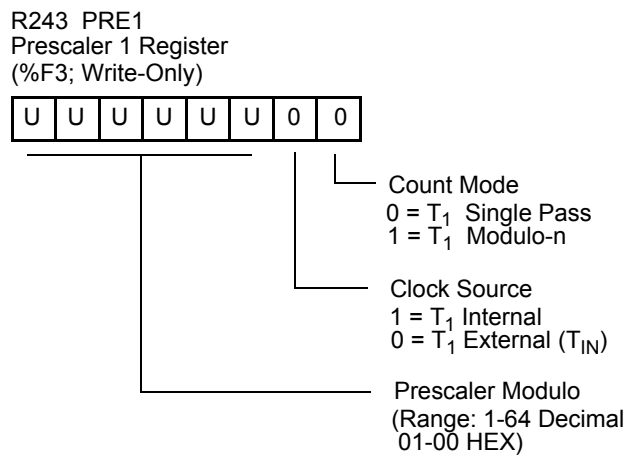
Figure 71. Prescaler 1 Register

R242 T1
Counter/Timer 1 Register
(%F2; Write/Read Only)

R244 T0
Counter/Timer 0 Register
(%F4; Write/Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

Initial value when written
(Range 1-256 decimal, 01-00 HEX)
current value when read
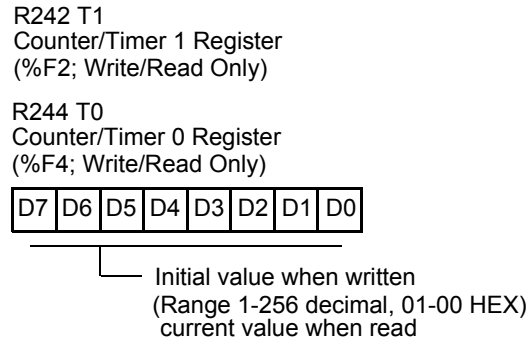
Figure 72. Counter/Timer 0 and 1 Registers

Counter/Timer Operation

Under software control, counter/timers are started and stopped via the Timer Mode Register (TMR,F1h) bits D0–D3 (see Figure 73). Each counter/timer is associated with a Load bit and an Enable Count bit.

Load and Enable Count Bits

Setting the Load bit (D0 for T0 and D2 for T1) transfers the initial value in the prescaler and the counter/timer registers into their respective down-counters. The next internal clock resets bits D0 and D2 to 0, readying the Load bit for the next load operation. New values may be loaded into the down-counters at any time. If the counter/timer is running, it continues to do so and starts the count over with the new value. Therefore, the Load bit actually functions as a software re-trigger.

R241 TMR
Timer Mode Register
(% F1; Read/Write)

				D3	D2	D1	D0

0 = No Function
1 = Load T_0

0 = Disable T_0 Count
1 = Enable T_0 Count

0 = No Function
1 = Load T_1

0 = Disable T_1 Count
1 = Enable T_1 Count
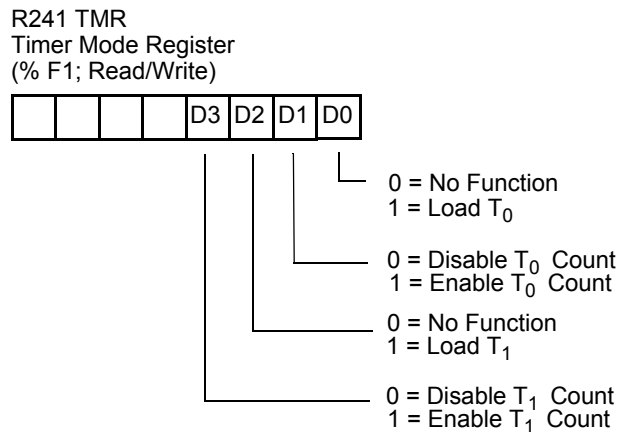
Figure 73. Timer Mode Register

The counter timers remain at rest as long as the Enable Count bits are 0. To enable counting, the Enable Count bit (D_1 for T0 and D_3 for T1) must be set to 1. Counting actually starts when the Enable Count bit is written by an instruction. The first decrement occurs four internal clock periods after the Enable Count bit has been set. If T1 is configured to use an external clock, the first decrement begins on the next clock period. The Load and Enable Count bits can be set at the same time. For example, using the instruction:

```
OR TMR,#03h
```

sets both D0 and D1 of the TMR. This loads the initial values of PRE0 and T0 into their respective counters and starts the count after the M2T2 machine state after the operand is fetched (see Figure 74).

R243 PRE1
Prescaler 1 Register
(% F3; Write-Only)

R245 PRE0
Prescaler 0 Register
(% F5; Write-Only)

Count Mode
0 = T_1 Single Pass
1 = T_1 Modulo-n
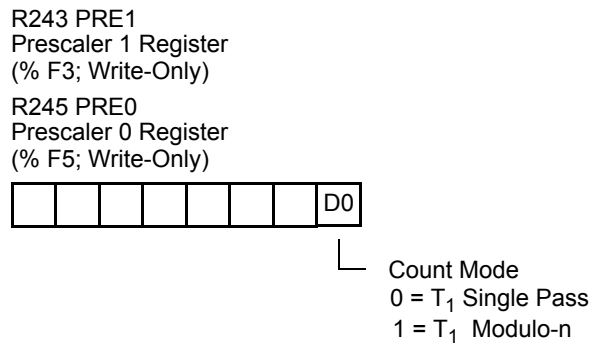
Figure 74. Starting The Count

First Decrement Occurs
Four Clock Periods Later

TMR is Written, Counter/Timer
is Loaded

#03h is Fetched
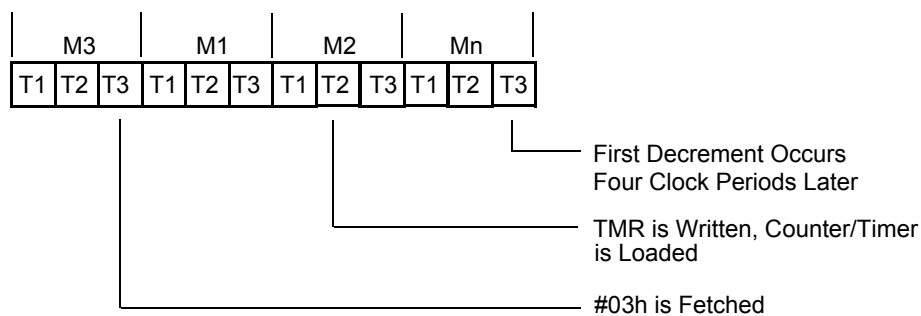
Figure 75. Counting Modes

Prescaler Operations

During counting, the programmed clock source drives the 6-bit Prescaler Counter. The counter is counted down from the value specified by bits of the corresponding Prescaler Register, PRE0 (bit 7 to bit 2) or PRE1 (bit 7 to bit 2; see Figure 70 and Figure 71 on page

82). When the Prescaler Counter reaches its end-of-count, the initial value is reloaded and counting continues. The prescaler never actually reaches 0. For example, if the prescaler is set to divide-by-three, the count sequence is:

3–2–1–3–2–1–3–2–1–3...

Each time the prescaler reaches its end of count a carry is generated, that allows the Counter/Timer to decrement by one on the next timer clock input. When the Counter/Timer and the prescaler both reach the end-of-count, an interrupt request is generated (IRQ4 for T0, IRQ5 for T1). Depending on the counting mode selected, the Counter/Timer either comes to rest with its value at `00h` (SINGLE-PASS Mode) or the initial value is automatically reloaded and counting continues (Continuous Mode). The counting modes are controlled by bit 0 of PRE0 and bit 0 of PRE1 (see Figure 75 on page 84). A 0, written to this bit configures the counter for Single-pass counting mode, while a 1 written to this bit configures the counter for Continuous mode.

The Counter/Timer can be stopped at any time by setting the Enable Count bit to 0, and restarted by setting it back to 1. The Counter/Timer continues its count value at the time it was stopped. The current value in the Counter/Timer can be read at any time without affecting the counting operation.

▶ **Note:** *The prescaler registers are write-only and cannot be read.*

New initial values can be written to the prescaler or the Counter/Timer registers at any time. These values are transferred to their respective down counters on the next load operation. If the Counter/Timer mode is continuous, the next load occurs on the timer clock following an end-of-count. New initial values should be written before the load operation, because the prescalers always effectively operate in Continuous count mode.

The time interval (i) until end-of-count, is given by the equation:

$i = t \, x \, p \, x \, v$

in which **t** = four times the internal clock period.

The internal clock frequency defaults to the external clock source (XTAL, ceramic resonator, and others) divided by 2. Some Z8 microcontrollers allow this divisor to be changed via the Stop Mode Recovery register. See the product data sheet for available clock divisor options.

t is equal to eight divided-by-XTAL frequency of the external clock source for T1 (external clock mode only).

p = the prescaler value (1–63) for T_0 and T_1.

The minimum prescaler count of 1 is achieved by loading `000001xx`. The maximum prescaler count of 63 is achieved by loading `111111xx`.

v = the Counter/Timer value (1–256)

Minimum duration is achieved by loading `01h` (1 prescaler output count), maximum duration is achieved by loading `00h` (256 prescaler outputs counts).

The prescaler and counter/timer are true divide-by-n counters.

T$_{OUT}$ Modes

The Timer Mode Register TMR (`F1h`; see Figure 76), is used in conjunction with the Port 3 Mode Register P3M (`F7h`; see Figure 77) to configure P36 for T$_{OUT}$ operation for T0 and T1. In order for T$_{OUT}$ to function, P36 must be defined as an output line by setting P3M bit 5 to 0. Output is controlled by one of the counter/timers (T0 or T1) or the internal clock.

Register F1hR
Timer Mode Register (TMR)
(Read/Write)

D7	D6			D3			D0

0 = No Function
1 = Load T$_0$

0 = Disable T$_1$ Count
1 = Enable T$_1$ Count

T$_{OUT}$ Modes:
T$_{OUT}$ OFF = 00
T$_0$ OUT = 01
T$_1$ OUT = 10
Internal Clock OUT = 11
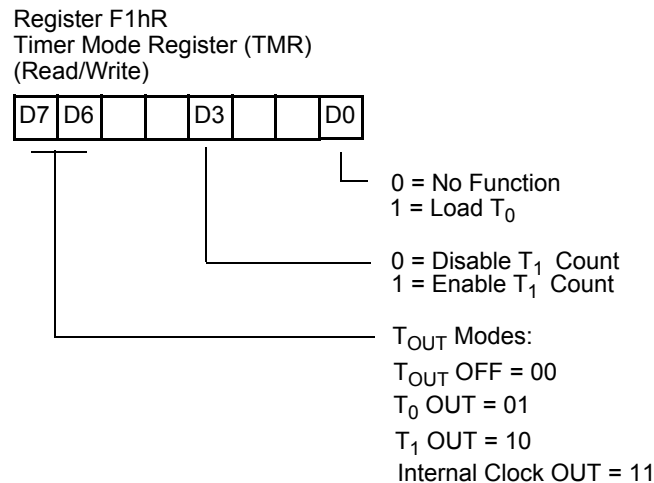
Figure 76. Timer Mode Register (T$_{OUT}$ Operation)

Register F7h
Port 3 Mode Register (P3M)
(Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

0 P31 = Input (T$_{IN}$)   P36 = Output (T$_{OUT}$)
1 P31 = DAV2/RDY2   P36 = RDY2/DAV2
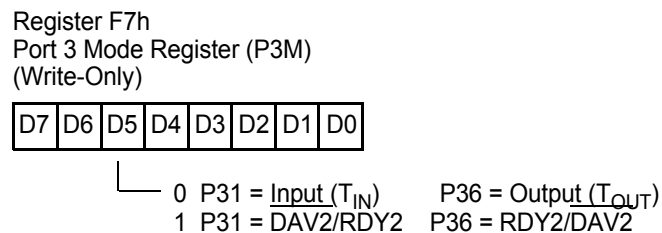
Figure 77. Port 3 Mode Register (T$_{OUT}$ Operation)

The counter/timer to be output is selected by TMR bit 7 and bit 6. T0 is selected to drive the T$_{OUT}$ line by setting bit 7 to 0 and bit 6 to 1. Likewise, T1 is selected by setting bit 7

and bit 6 to 1 and 0, respectively. The counter/timer T_{OUT} mode is turned off by setting TMR bit and bit 6 both to 0, freeing P36 to be a data output line.

T_{OUT} is initialized to a logic 1 whenever the TMR Load bit (bit 0 for T0 or bit 1 for T2) is set to 1. The T_{OUT} configuration timer load, and Timer Enable Count bits for the counter/timer driving the T_{OUT} pin can be set at the same time. For example, using the instruction:
`OR TMR,#43h`

- Configures T0 to drive the T_{OUT} pin (P36)

- Sets the P36 T_{OUT} pin to a logic 1 level

- Loads the initial PRE0 and T0 levels into their respective counters and starts the counter after the M2T2 machine state after the operand is fetched

At end-of-count, the interrupt request line (IRQ4 or IRQ5), clocks a toggle flip-flop. The output of this flip-flop drives the T_{OUT} line, P36. In all cases, when the selected counter/timer reaches its end-of-count, T_{OUT} toggles to its opposite state (see Figure 78). If, for example, the counter/timer is in CONTINUOUS COUNTING Mode, Tout has a 50 percent duty cycle output. This duty cycle can easily be controlled by varying the initial values after each end-of-count.

The internal clock can be selected as output instead of T0 or T1 by setting TMR bit 7 and bit 6 both to 1. The internal clock (XTAL frequency ÷ 2) is then directly output on P36 (see Figure 79 on page 88).

While programmed as T_{OUT}, P36 cannot be modified by a write to port register P3. However, the Z8® software can examine the P36 current output by reading the port register.
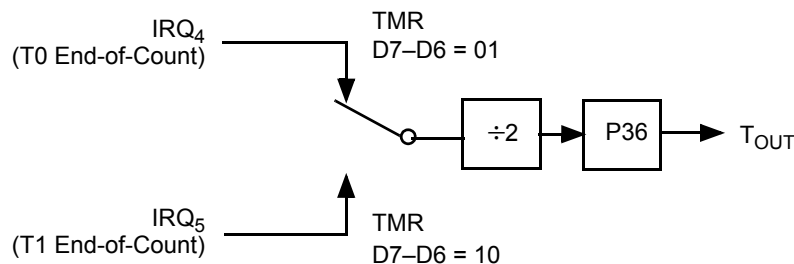
Figure 78. T0 and T1 Output Through T_{OUT}
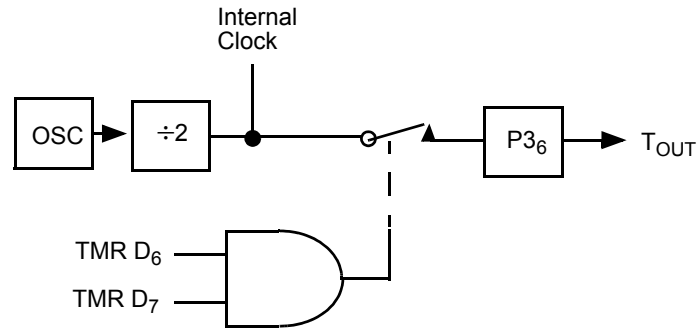
Figure 79. Internal Clock Output Through T$_{OUT}$

T$_{IN}$ Modes

The Timer Mode Register TMR (F1h; see Figure 80 on page 89) is used in conjunction with the Prescaler Register PRE1 (F3h; see Figure 81 on page 89) to configure P31 as T$_{IN}$. T$_{IN}$ is used in conjunction with T1 in one of four modes:

- External Clock Input
- Gated Internal Clock
- Triggered Internal Clock
- Retriggerable Internal Clock

> **Note:** *The T$_{IN}$ mode is restricted for use with timer 1 only. To enable the T$_{IN}$ mode selected (via TMR bits 4-5), bit 1 of PRE1 must be set to 0.*

The counter/timer clock source must be configured for external by setting the PRE1 Register bit 2 to 1. The Timer Mode Register bit 5 and bit 4 can then be used to select the appropriate T$_{IN}$ operation.

For T1 to start counting as a result of a T$_{IN}$ input, the Enable Count bit (bit 3 in TMR) must be set to 1. When using T$_{IN}$ as an external clock or a gate input, the initial values must be loaded into the down counters by setting the Load bit (bit 2 in TMR) to a 1 before counting begins. In the descriptions of T$_{IN}$ that follow, it is assumed the programmer has performed these operations. Initial values are automatically loaded in Trigger and Retrigger modes so software loading is unnecessary.

Register F1h
Timer Mode Register (TMR)
(Read/Write)

| | | D5 | D4 | | | | |

T_{IN} = Modes:
External Clock Input = 00
Gate Input = 01
Trigger Input = 10 (Non-retriggerable)
Trigger Input = 11 (Retriggerable)
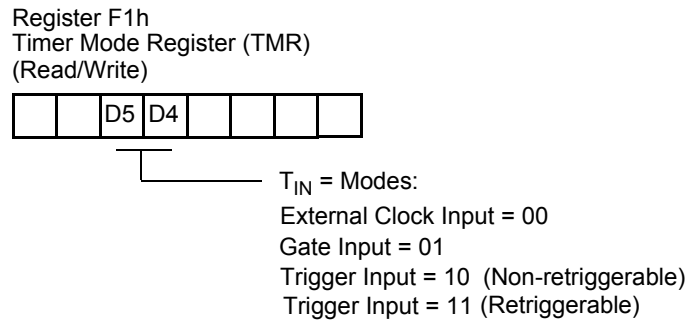
Figure 80. Timer Mode Register (T_{IN} Operation)

Register F3h
Prescaler 1 Register (PRE1)
(Write-Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Clock Source
0 = T_1 External   Enable T_{IN} Mode
1 = T_1 Internal   Disable T_{IN} Mode
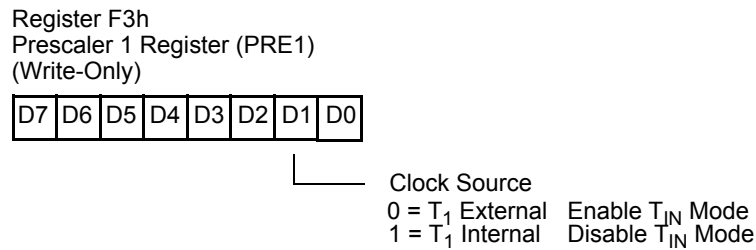
Figure 81. Prescaler 1 Register (T_{IN} Operation)

It is suggested that P31 be configured as an input line by setting P3M Register bit 5 to 0, although T_{IN} is still functional if P31 is configured as a handshake input.

Each High-to-Low transition on T_{IN} generates an interrupt request IRQ2, regardless of the selected T_{IN} mode or the enabled/disabled state of T1. IRQ2 must therefore be masked or enabled according to the requirements of the application.

External Clock Input Mode

The T_{IN} External Clock Input Mode (TMR bit 5 and bit 4 both set to 0) supports counting of external events, where an event is considered to be a High-to-Low transition on T_{IN} (see Figure 82).

Note: *See the product data sheet for the minimum allowed T_{IN} external clock input period (T_P T_{IN}).*
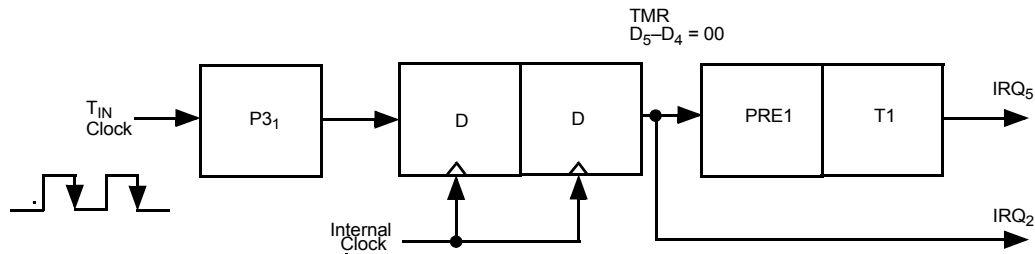
Figure 82. External Clock Input Mode

Gated Internal Clock Mode

The T_{IN} Gated Internal Clock Mode (TMR bit 5 and bit 4 set to 0 and 1 respectively) measures the duration of an external event. In this mode, the T1 prescaler is driven by the internal timer clock, gated by a High level on T_{IN} (see Figure 83). T1 counts while T_{IN} is High and stops counting while T_{IN} is Low. Interrupt request IRQ2 is generated on the High-to-Low transition of T_{IN} signalling the end of the gate input. Interrupt request IRQ5 is generated if T1 reaches its end-of-count.
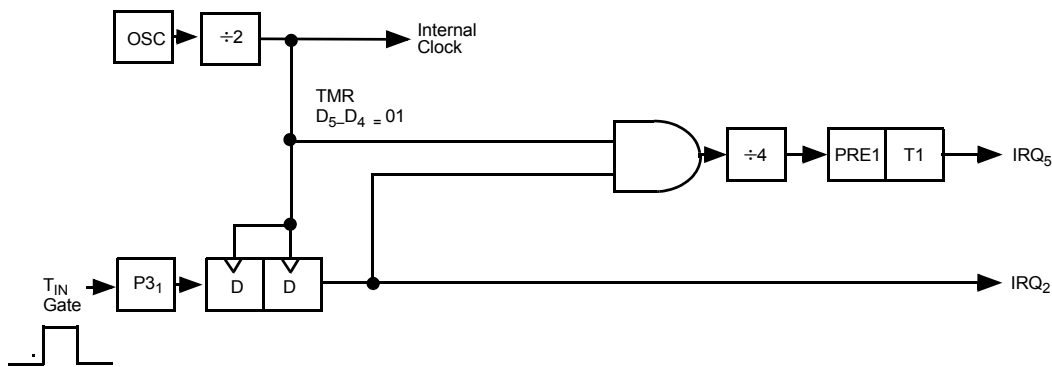
Figure 83. Gated Clock Input Mode

Triggered Input Mode

The T_{IN} Triggered Input Mode (TMR bits 5 and 4 are set to 1 and 0, respectively) causes T1 to start counting as the result of an external event (see Figure 84 on page 91). T1 is then loaded and clocked by the internal timer clock following the first High-to-Low transition on the T_{IN} input. Subsequent T_{IN} transitions do not affect T1. In SINGLE-PASS mode, the Enable bit is reset whenever T1 reaches its end-of-count. Further T_{IN} transitions have no effect on T1 until software sets the Enable Count bit again. In CONTINUOUS mode, once

T1 is triggered counting continues until software resets the Enable Count bit. Interrupt request IRQ5 is generated when T1 reaches its end-of-count.
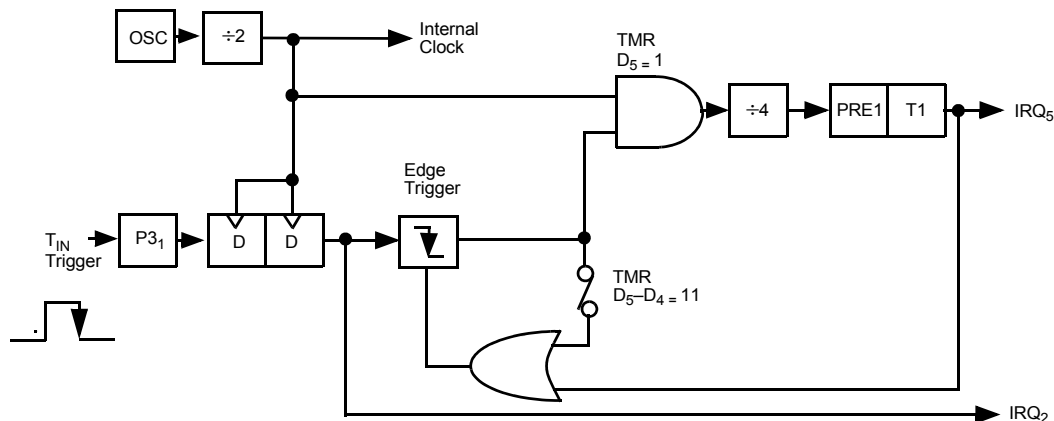
Figure 84. Triggered Clock Mode

Retriggerable Input Mode

The T_{IN} Retriggerable Input Mode (TMR bits 5 and 4 are set to 1) causes T1 to load and start counting on every occurrence of a High-to-Low transition on T_{IN}, see Figure 84. Interrupt request IRQ5 is generated if the programmed time interval (determined by T1 prescaler and counter/timer register initial values) has elapsed because of the last High-to-Low transition on T_{IN}. In SINGLE-PASS Mode, the end-of-count resets the Enable Count bit. Subsequent T_{IN} transitions do not cause T1 to load and start counting until software sets the Enable Count bit again. In Continuous Mode, counting continues once T1 is triggered until software resets the Enable Count bit. When enabled, each High-to-Low T_{IN} transition causes T1 to reload and restart counting. Interrupt request IRQ5 is generated on every end-of-count.

Cascading Counter/Timers

For some applications, it may be necessary to measure a time interval greater than a single counter/timer can measure. In this case, T_{IN} and T_{OUT} can be used to cascade T0 and T1 as a single unit (see Figure 85 on page 92). T0 should be configured to operate in Continuous mode and to drive T_{OUT}. T_{IN} should be configured as an external clock input to T1 and wired back to T_{OUT}. On every other T0 end-of-count, T_{OUT} undergoes a High-to-Low transition that causes T1 to count.

T1 can operate in either Single-Pass or Continuous mode. When the T1 end-of-count is reached, interrupt request IRQ5 is generated. Interrupt requests IRQ2 (T_{IN} High-to-Low transitions) and IRQ4 (T0 end-of-count) are also generated but are most likely of no importance in this configuration and should be disabled.
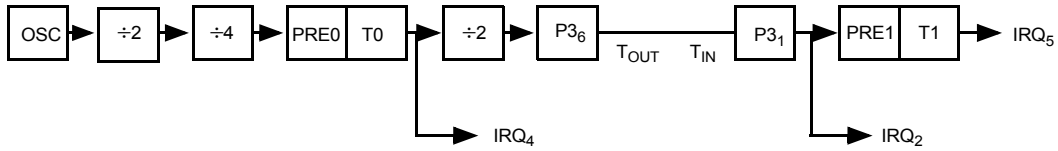
Figure 85. Cascaded Counter/Timers

Reset Conditions

After a hardware reset, the counter/timers are disabled and the contents of the counter/timer and prescaler registers are undefined. However, the counting modes are configured for Single-Pass and the T1 clock source is set for external.

R242 T1
Counter/Timer 1 Register
(%F2; Write/Read Only)

R244 T0
Counter/Timer 0 Register
(%F4; Write/Read Only)

U	U	U	U	U	U	U	U

Initial value when written
(Range 1–256 decimal, 01–00 HEX)
current value when read

Figure 86. Counter/Timer Reset

T_{IN} is set for External Clock mode, and the T_{OUT} mode is OFF. Figure 87 on page 93 through Figure 89 on page 94 displays the binary reset values of the Prescaler, Counter/Timer, and Timer Mode registers.

R243 PRE1
PRESCALER 1 REGISTER
(%F3; WRITE-ONLY)

U	U	U	U	U	U	0	0

COUNT MODE
$0 = T_1$ SINGLE PASS
$1 = T_1$ MODULO-N

CLOCK SOURCE
$1 = T_1$ INTERNAL
$0 = T_1$ EXTERNAL (T_{IN})

PRESCALER MODULO
(RANGE: 1–64 DECIMAL
 01–00 HEX)
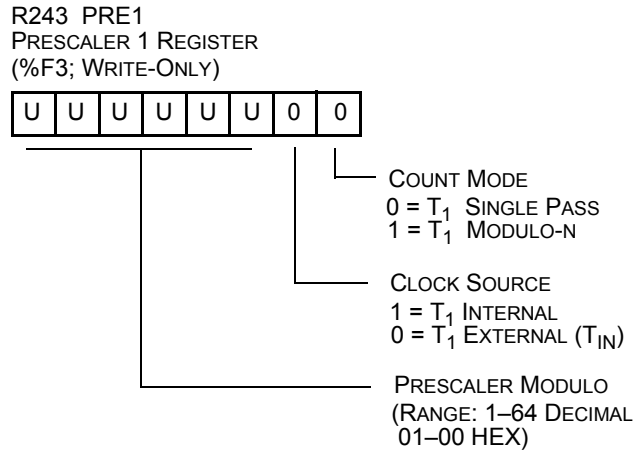
Figure 87. Prescaler 1 Register Reset

R245 PRE0
Prescaler 0 Register
(%F5; Write-Only)

U	U	U	U	U	U	U	0

Count Mode
$0 = T_0$ Single Pass
$1 = T_0$ Modulo-n

Reserved (Must be 0)

Prescaler Modulo
(Range: 1–64 Decimal
 01–00 HEX)
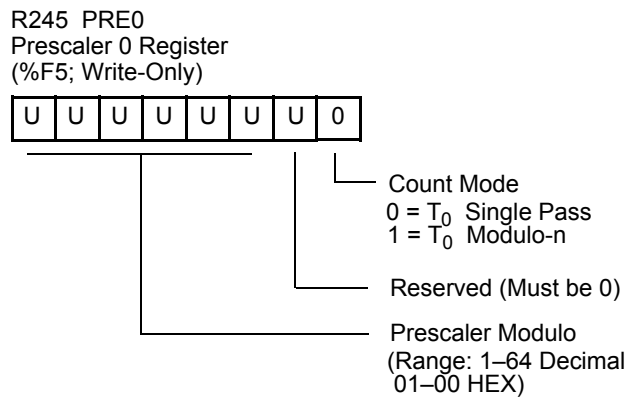
Figure 88. Prescaler 0 Reset

R241 TMR
Timer Mode Register
(% F1; Read/Write)

0	0	0	0	0	0	0	0

0 = No Function
1 = Load T_0

0 = Disable T_0 Count
1 = Enable T_0 Count

0 = No Function
1 = Load T_1

0 = Disable T_1 Count
1 = Enable T_1 Count

T_{IN} = Modes:

External Clock Input = 00

Gate Input = 01

Trigger Input = 10
 (Non-retriggerable)

Trigger Input = 11
(Retriggerable)

T_{OUT} Modes:
T_{OUT} OFF = 00
T_0 OUT = 01
T_1 OUT = 10
 Internal Clock OUT = 11
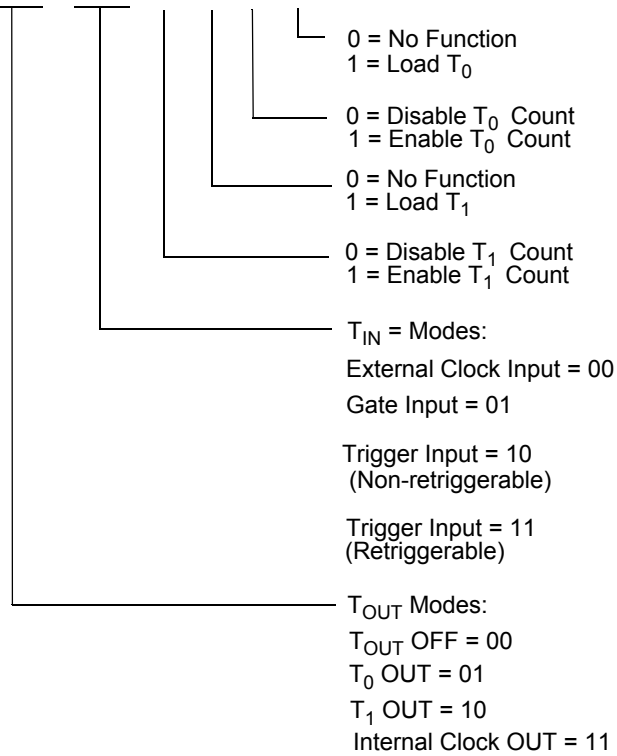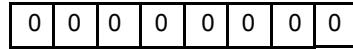
Figure 89. Timer Mode Register Reset

Interrupts

Z8® CPU allows 6 different interrupts from a variety of sources; up to four external inputs, the on-chip Counter/Timer(s), software, and serial I/O peripherals. These interrupts can be masked and their priorities are set by using the Interrupt Mask and the Interrupt Priority Registers. All six interrupts can be globally disabled by resetting the master Interrupt Enable, bit 7 in the Interrupt Mask Register, with a Disable Interrupt (DI) instruction. Interrupts are globally enabled by setting bit 7 with an Enable Interrupt (EI) instruction.

Register	HEX	Identifier
7		
Interrupt Mask	FBh	IMR
Interrupt Request	FAh	IRQ
Interrupt Priority	F9h	IPR

Figure 90. Interrupt Control Registers

There are three interrupt control registers: the Interrupt Request Register (IRQ), the Interrupt Mask register (IMR), and the Interrupt Priority Register (IPR). Figure 90 displays addresses and identifiers for the interrupt control registers. Figure 91 on page 96 displays the block diagram illustrating the Interrupt Mask and Interrupt Priority logic. The Z8 family supports both vectored and polled interrupt handling.
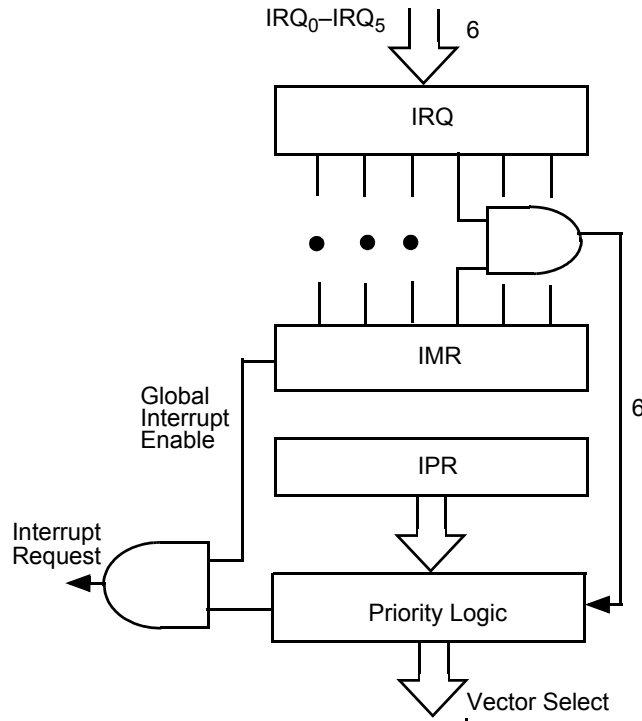
Figure 91. Interrupt Block Diagram

> **Note:** *See the selected Z8 CPU's product specification for the exact interrupt sources supported.*

Interrupt Sources

Table 18 on page 97 describes the interrupt types, sources, and vectors available in the Z8® family of processors.

Table 18. Interrupt Types, Sources, and Vectors

Name	Sources	Vector Location	Comments
IRQ$_0$	DAV0, IRQ0, Comparator	0,1	External (P32), Edge Triggered; Internal
IRQ$_1$	DAV1, IRQ1	2,3	External (P33), Edge Triggered; Internal
IRQ$_2$	DAV2, IRQ2, T$_{IN}$, Comparator	4,5	External (P31), Edge Triggered; Internal
	IRQ3	6,7	External (P30) or (P32), Edge Triggered; Internal
	Serial In	6,7	Internal
	T0	8,9	Internal
	Serial Out	8,9	Internal
IRQ$_5$	T1	10,11	Internal

External Interrupt Sources

External sources involve interrupt request lines IRQ0–IRQ3. IRQ0, IRQ1, and IRQ2 can be generated by a transition on the corresponding Port 3 pin (P32, P33, and P31 correspond to IRQ0, IRQ1, and IRQ2, respectively).

Figure 92 displays the block diagram for interrupt sources IRQ0, IRQ1, and IRQ2.

▶ **Note:** *The interrupt sources and trigger conditions are device dependent. See the device product specification to determine available sources (internal and external), triggering edge options, and exact programming details.*
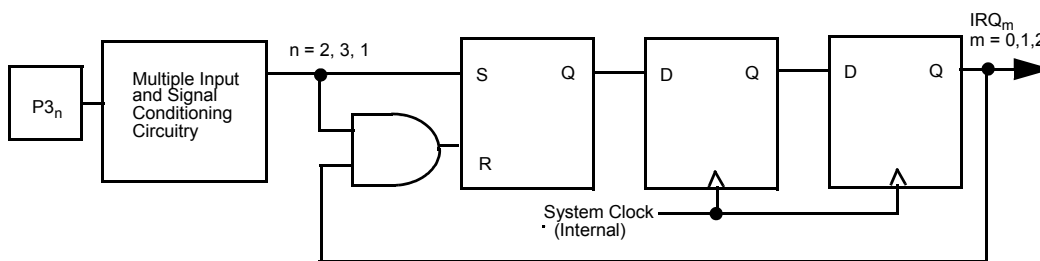
Figure 92. Interrupt Sources IRQ0–IRQ2 Block Diagram

When the Port 3 pin (P31, P32, or P33) transitions, the first flip-flop is set. The next two flip-flops synchronize the request to the internal clock and delay it by two internal clock periods. The output of the last flip-flop (IRQ0, IRQ1, or IRQ2) goes to the corresponding Interrupt Request Register.

IRQ3 can be generated from an external source only if Serial In is not enabled. Otherwise, its source is internal. The external request is generated by a Low edge signal on P30 as displayed in Figure 93. Again, the external request is synchronized and delayed before reaching IRQ3. Some Z8 products replace P30 with P32 as the external source for IRQ3. In this case, IRQ3 interrupt generation follows the logic as displayed in Figure 92 on page 97.

> **Note:** *Although interrupts are edge triggered, minimum interrupt request Low and High times must be observed for proper operation. See the device product specification for exact timing requirements on external interrupt requests ($T_W IL$, $T_W IH$).*
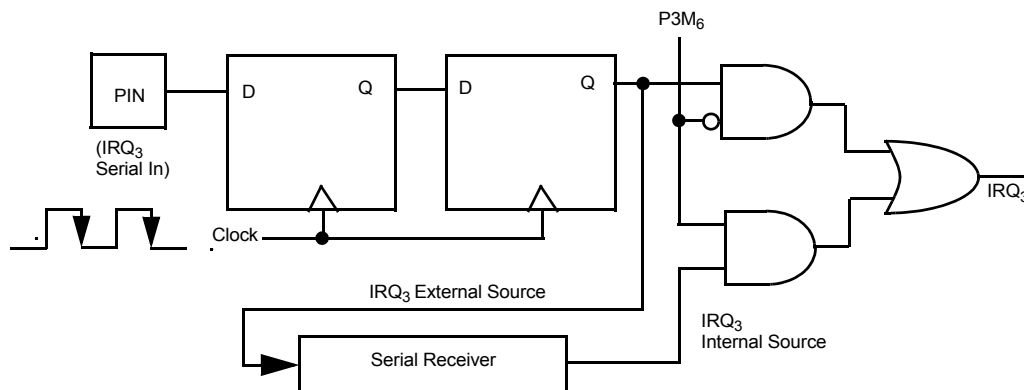
Figure 93. Interrupt Source IRQ3 Block Diagram

Internal Interrupt Sources

Internal sources involve interrupt requests IRQ0, IRQ2, IRQ3, IRQ4, and IRQ5. Internal sources are ORed with the external sources, so either an internal or external source can trigger the interrupt. Internal interrupt sources and trigger conditions are device dependent.

Refer to the device product specification to determine available sources, triggering edge options, and exact programming details. For more details on the internal interrupt sources, see Counters and Timers, Input/Output Ports, and Serial Input/Output sections.

Interrupt Request Register Logic and Timing

Figure 94 displays the logic diagram for the Interrupt Request (IRQ) Register. The leading edge of the request sets the first flip-flop, that remains set until interrupt requests are sampled.

Requests are sampled internally during the last clock cycle before an opcode fetch (see Figure 95). External requests are sampled two internal clocks earlier, due to the synchronizing flip-flops as displayed in Figure 92 on page 97 and Figure 93 on page 98.

At sample time the request is transferred to the second flip-flop in Figure 94, that drives the interrupt mask and priority logic. When an interrupt cycle occurs, this flip-flop is reset only for the highest priority level that is enabled.

You have direct access to the second flip-flop by reading and writing the IRQ Register. IRQ is read by specifying it as the source register of an instruction and written by specifying it as the destination register.
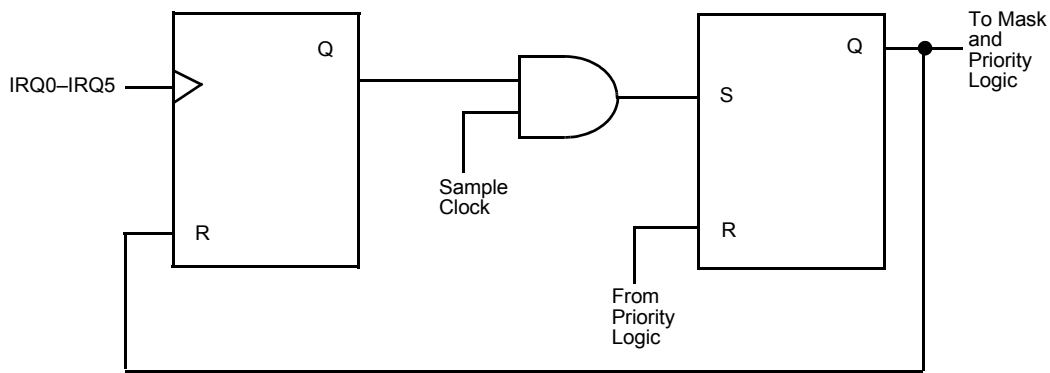
Figure 94. IRQ Register Logic
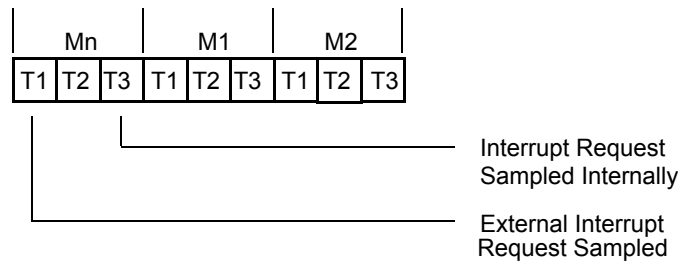
Figure 95. Interrupt Request Timing

Interrupt Initialization

After reset, all interrupts are disabled and must be initialized before vectored or polled interrupt processing can begin. The Interrupt Priority Register (IPR), Interrupt Mask Register (IMR), and Interrupt Request Register (IRQ) must be initialized, in that order, to start the interrupt process.

Interrupt Priority Register Initialization

The Interrupt Priority Register (IPR) displayed in Figure 96 is a write-only register that sets priorities for the vectored interrupts in order to resolve simultaneous interrupt requests. (There are 48 sequence possibilities for interrupts.) The six interrupt levels IRQ0-IRQ5 are divided into three groups of two interrupt requests each. One group contains IRQ3 and IRQ5. The second group contains IRQ0 and IRQ2, while the third group contains IRQ1 and IRQ4.

Priorities can be set both within and between groups as listed in Table 19 on page 100 and Table 20 on page 101. Bits 1, 2, and 5 define the priority of the individual members within the three groups. Bits 0, 3, and 4 are encoded to define six priority orders among the three groups. Bits 6 and 7 are reserved.
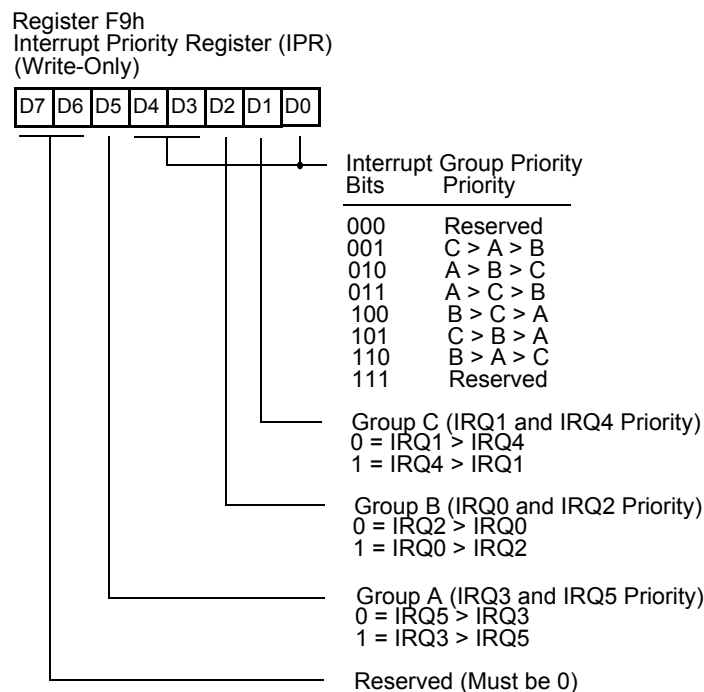
Figure 96. Interrupt Priority Register

Table 19. Interrupt Priority

Group	Bit	Value	Priority	
			Highest	**Lowest**
C	Bit 1	0	IRQ1	IRQ4
		1	IRQ4	IRQ1

Table 19. Interrupt Priority (Continued)

Group	Bit	Value	Priority Highest	Priority Lowest
B	Bit 2	0	IRQ2	IRQ0
		1	IRQ0	IRQ2
A	Bit 5	0	IRQ5	IRQ3
		1	IRQ3	IRQ5

Table 20. Interrupt Group Priority

Bit 4	Bit 3	Bit 0	High	Medium	Low
0	0	0	Not Used		
0	0	1	C	A	B
0	1	0	A	B	C
0	1	1	A	C	B
1	0	0	B	C	A
1	0	1	C	B	A
1	1	0	B	A	C
1	1	1	Not Used		

Interrupt Mask Register Initialization

The Interrupt Mask Register individually or globally enables or disables the six interrupt requests (see Figure 97 on page 102). When bit 0 to bit 5 are set to 1, the corresponding interrupt requests are enabled. Bit 7 is the master enable and must be set before any of the individual interrupt requests can be recognized. Resetting bit 7 globally disables all of the interrupt requests. Bit 7 is set and reset by the EI and DI instructions. It is automatically reset during an interrupt service routine and set following the execution of an Interrupt Return (IRET) instruction.

Bit 7 must be reset by the DI instruction before the contents of the Interrupt Mask Register or the Interrupt Priority Register are changed except:

- Immediately after a hardware reset

- Immediately after executing an interrupt service routine and before IMR bit 7 has been set by any instruction

Register FBh
Interrupt Request Register (IMR)
(Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

0 = Disables IRQ0
1 = Enables IRQ0

0 = Disables IRQ1
1 = Enables IRQ1

0 = Disables IRQ2
1 = Enables IRQ2

0 = Disables IRQ3
1 = Enables IRQ3

0 = Disables IRQ4
1 = Enables IRQ4

0 = Disables IRQ5
1 = Enables IRQ5

0 = Disables RAM Protect
1 = Enables RAM Protect

0 = Disables Interrupt
1 = Enables Interrupt
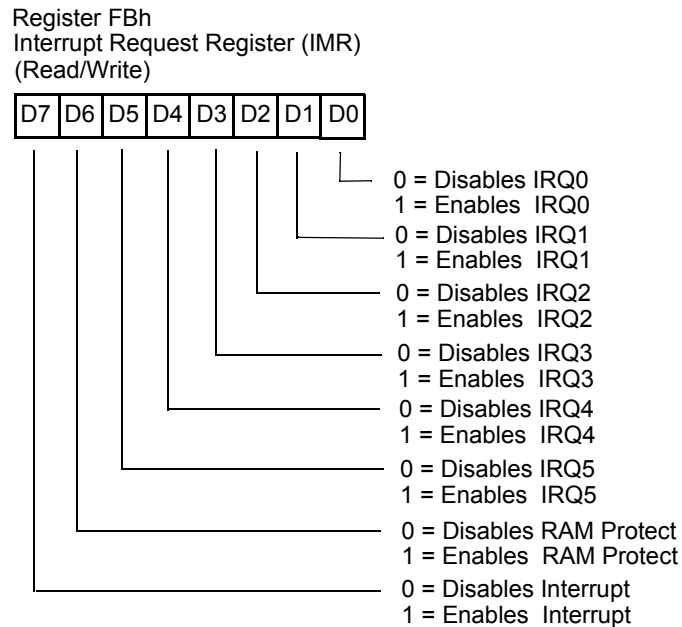
Figure 97. Interrupt Mask Register

The RAM Protect option is selected at ROM mask submission time or at EPROM program time. If not selected or not an available option, this bit is reserved and must be 0.

Interrupt Request Register Initialization

The Interrupt Request Register (IRQ), displayed in Figure 98 on page 103, is a read/write register that stores the interrupt requests for both vectored and polled interrupts. When an interrupt is made on any of the six, the corresponding bit position in the register is set to 1. Bit 0 to bit 5 are assigned to interrupt requests IRQ0 to IRQ5, respectively.

Whenever Power-On Reset (POR) is executed, the IRQ resister is reset to `00h` and disabled. Before the IRQ register accepts requests, it must be enabled by executing an ENABLE INTERRUPTS (EI) instruction.

> **Note:** *Setting the Global Interrupt Enable bit in the Interrupt Mask Register (IMR, bit 7) does not enable the IRQ. Execution of the EI instruction is required (see Figure 99 on page 104).*

For polled processing, IRQ must still be initialized by an EI instruction. To properly initialize the IRQ register, the following code is provided.

```
CLR    IMR    // Make sure vectored interrupts are disabled.
```

EI   // Enable IRQ register, otherwise read only.
       // Not required if interrupts were previously enabled.

DI   // Disable interrupt heading.

> **Note:** *IRQ is always cleared to* 00h *and is read only until the first EI instruction, which enables the IRQ to be read/written.*

Register FAh
Interrupt Request Register (IRQ)
(Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

0 = IRQ0 RESET
1 = IRQ0 SET

0 = IRQ1 RESET
1 = IRQ1 SET

0 = IRQ2 RESET
1 = IRQ2 SET

0 = IRQ3 RESET
1 = IRQ3 SET

0 = IRQ4 RESET
1 = IRQ4 SET

0 = IRQ5 RESET
1 = IRQ5 SET

Reserved /Int Edge Select
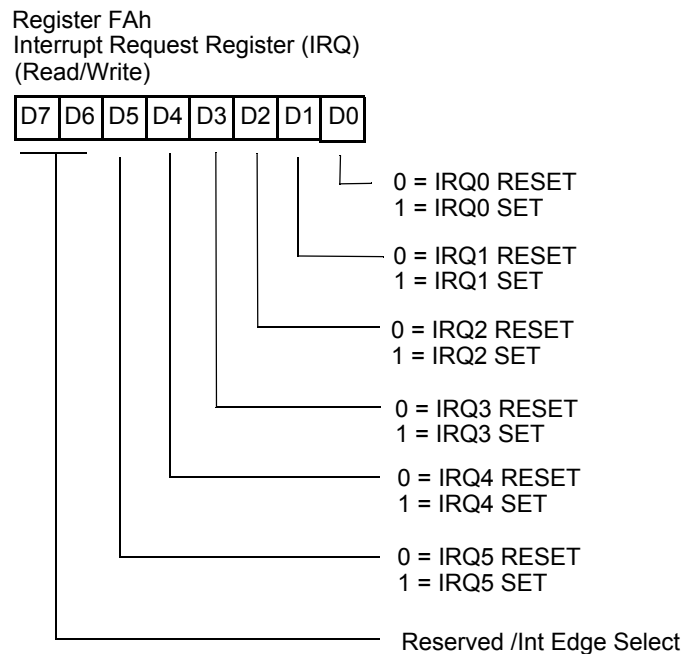
Figure 98. Interrupt Request Register

IMR is cleared before the IRQ enabling sequence to insure no unexpected interrupts occur when EI is executed. This code sequence should be executed prior to programming the application required values for IPR and IMR.

> **Note:** *IRQ bits 6 and 7 are device dependent. When reserved, the bits are not used and return a 0 when read. When used as the Interrupt Edge select bits, the configuration options are as listed in* Table 21 *on page 104.*

Table 21. IRQ Register Configuration*

IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

*F = Falling Edge; R = Rising Edge.

The proper sequence for programming the interrupt edge select bits is (assumes IPR and IMR have been previously initialized).

DI		;Inhibit all interrupts until input edges are configured.
OR	IRQ,#XX 000000b	;Configure interrupt; do not disturb edges as required—IRQ 0-5.
EI		;Re-enable interrupts.
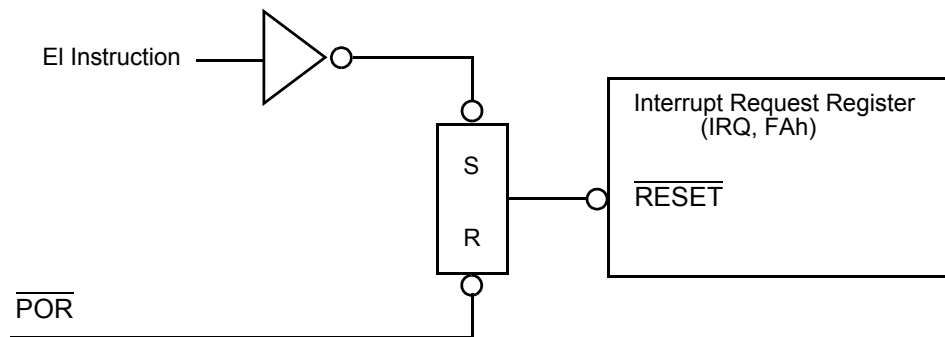
Figure 99. IRQ Reset Functional Logic Diagram

IRQ Software Interrupt Generation

IRQ can be used to generate software interrupts by specifying IRQ as the destination of any instruction referencing the Z8® Standard Register File. These Software Interrupts (SWI) are controlled in the same manner as hardware generated requests (in other words, the IPR and the IMR control the priority and enabling of each SWI level).

To generate a SWI, the appropriate request bit in the IRQ is set as follows:

```
ORIRQ, #NUMBER
```

where the immediate data, NUMBER, has a 1 in the bit position corresponding to the appropriate level of the SWI. For example, if an SWI is required on IRQ5, NUMBER would have a 1 in bit 5:

```
OR IRQ, #00100000b
```

With this instruction, if the interrupt system is globally enabled, IRQ5 is enabled, and there are no higher priority pending requests, control is transferred to the service routine pointed to by the IRQ5 vector.

Vectored Processing

Each Z8 interrupt level has its own vector. When an interrupt occurs, control passes to the service routine pointed to by the interrupt's vector location in Program Memory. The sequence of events for vectored interrupts is as follows:

- PUSH PC Low byte on stack

- PUSH PC High byte on stack

- PUSH FLAGS on stack

- Fetch High byte of vector

- Fetch Low byte of vector

- Branch to service routine specified by vector

SP and Stack before an interrupt   SP and Stack after an interrupt
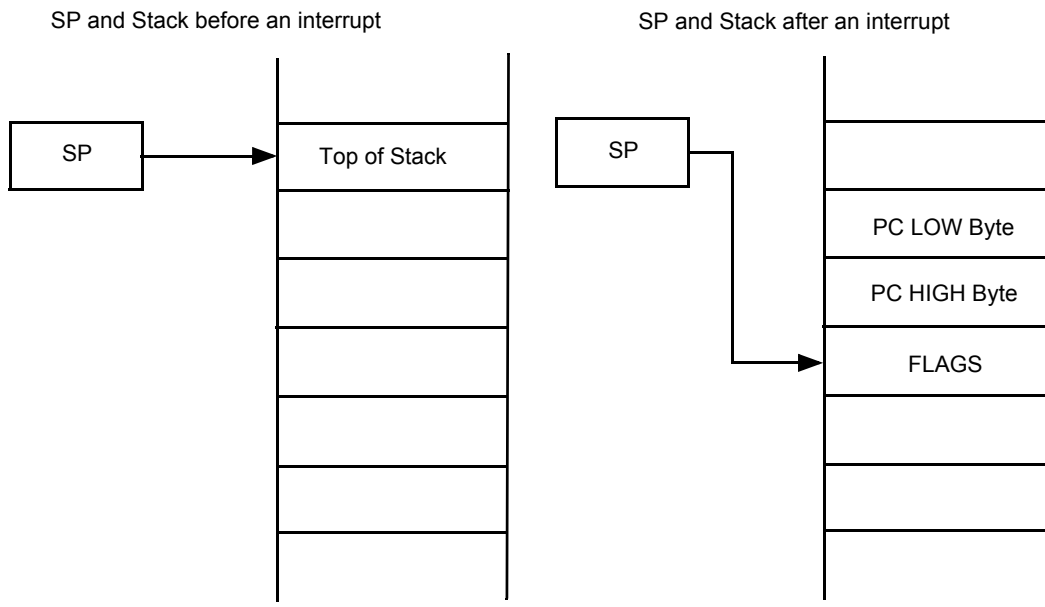
Figure 100. Effects of an Interrupt on the Stack
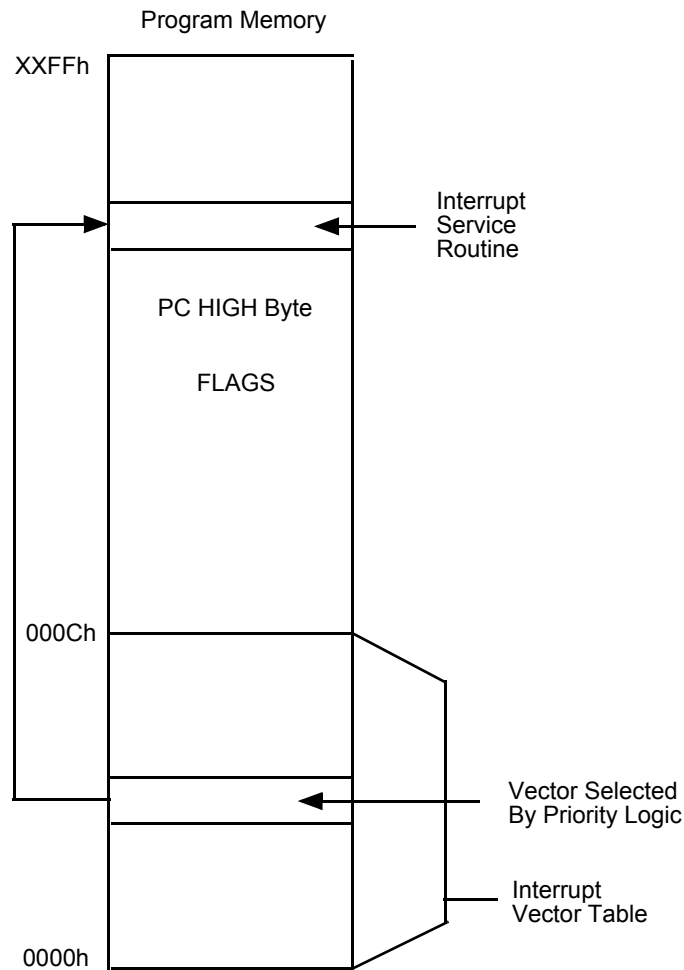
Figure 101. Interrupt Vectoring

Vectored Interrupt Cycle Timing

The interrupt acknowledge cycle time is 24 internal clock cycles and is displayed in Figure 102 on page 108. In addition, two internal clock cycles are required for the synchronizing flip-flops. The maximum interrupt recognition time is equal to the number of clock cycles required for the longest executing instruction present in the user program (assumes worst case condition of interrupt sampling, Figure 95 on page 99, just prior to the interrupt occurrence). To calculate the worst case interrupt latency (maximum time required from interrupt generation to fetch of the first instruction of the interrupt service routine), sum these components:

Worst Case Interrupt Latency ≈ 24 INT CLK (interrupt acknowledge time) + # T_PC of

longest instruction present in the application program + 2T$_P$C (internal synchronization time).
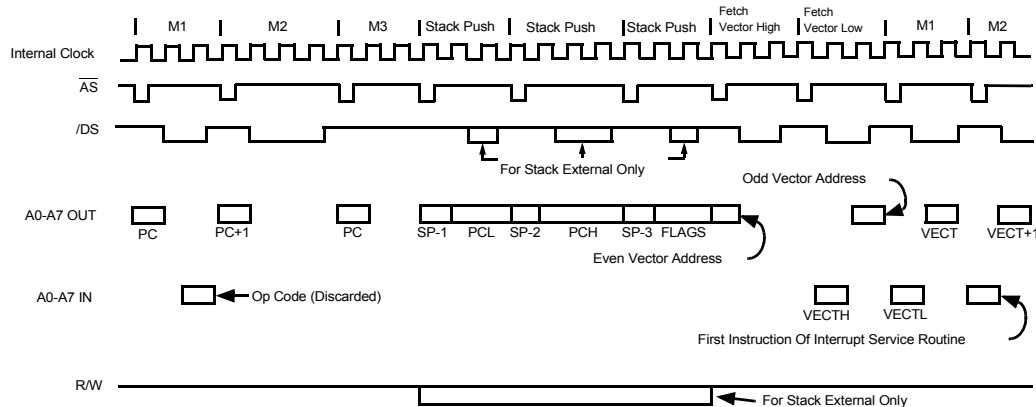
Figure 102. Z8 Interrupt Acknowledge Timing

Nesting of Vectored Interrupts

Nesting of vectored interrupts allows higher priority requests to interrupt a lower priority request. To initiate vectored interrupt nesting, do the following during the interrupt service routine:

- Push the old IMR on the stack

- Load IMR with a new mask to disable lower priority interrupts

- Execute EI instruction

- Proceed with interrupt processing

- After processing is complete, execute DI instruction

- Restore the IMR to its original value by returning the previous mask from the stack

- Execute IRET

Depending on the application, some simplification of the above procedure may be possible.

Polled Processing

Polled interrupt processing is supported by masking off the IRQ to be polled. This is accomplished by clearing the corresponding bits in the IMR.

To enable any interrupt, first the interrupt mechanism must be engaged with an EI instruction. If only polled interrupts are to be serviced, execute:

EI ;Enable interrupt mechanism
DI ;Disable vectored interrupts.

To initiate polled processing, check the bits of interest in the IRQ using the Test Under Mask (TM) instruction. If the bit is set, call or branch to the service routine. The service routine services the request, resets its Request Bit in the IRQ, and branches or returns back to the main program. An example of a polling routine is as follows:

```
        TM IRQ, #MASKA    ;Test for request
        JR Z, NEXT        ;If no request go to NEXT
        CALL SERVICE      ;If request is there, then
                          ;service it
NEXT:
    .
    .
    .
SERVICE:                  ;Process Request
    .
    .
    .
        AND IRQ, #MASKB   ;Clear Request Bit
        RET               ;Return to next
```

In this example, if IRQ2 is being polled, MASKA is `00000100b` and MASKB is `11111011b`.

Reset Conditions

Upon reset, all bits in IPR are undefined.

In IMR, bit 7 is 0 and bits 0–6 are undefined. The IRQ register is reset and held in that state until an enable interrupt (EI) instruction is executed.

Power-Down Modes

In addition to the standard RUN mode, the Z8$^®$ CPU supports two Power-Down modes to minimize device current consumption. The two modes supported are HALT and STOP.

Halt Mode Operation

HALT mode suspends instruction execution and turns off the internal CPU clock. The on-chip oscillator circuit remains active so the internal clock continues to run and is applied to the Counter/Timer(s) and interrupt logic.

To enter HALT mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the application program must execute a NOP instruction (opcode = `FFh`) immediately before the HALT instruction (opcode `7Fh`), that is:

```
FF NOP ;clear the instruction pipeline
7F HALT ;enter HALT mode
```

HALT mode is exited by interrupts, either externally or internally generated. Upon completion of the interrupt service routine, the user program continues from the instruction after HALT.

HALT mode may also be exited via a POR/RESET activation or a Watch–Dog Timer (WDT) time-out. (See the product data sheet for WDT availability). In this case, program execution restarts at the reset restart address `000Ch`.

To further reduce power consumption in HALT mode, some Z8 family devices allow dynamic internal clock scaling. Clock scaling may be accomplished on the fly by reprogramming bit 0 and/or bit1 of the Stop Mode Recovery register. See Figure 103 on page 112.

> **Note:** *Internal clock scaling directly affects Counter/Timer operation—adjustment of the prescaler and downcounter values may be required. To determine the actual HALT mode current (I_{CC1}) value for the various optional modes available, see the related Z8 device's product specification.*

Stop Mode Operation

STOP mode provides the lowest possible device standby current. This instruction turns OFF the on-chip oscillator and internal system clock.

To enter STOP mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the application program must execute a

NOP instruction (opcode = `FFh`) immediately before the STOP instruction
(opcode = `6Fh`), that is,

```
FF NOP ;clear the instruction pipeline
6F STOP ;enter STOP mode
```

STOP mode is exited by any one of the following resets: POR activation, WDT time out
(if available), or a Stop Mode Recovery source. Upon reset generation, the processor
always restarts the application program at address `000Ch`.

POR/RESET activation is present on all Z8 devices and is implemented as a reset pin and/
or an on-chip power on reset circuit.

Some Z8 devices allow for the on-chip WDT to run in STOP mode. If so activated, the
WDT time-out generates a reset some fixed time period after entering STOP mode.

Stop Mode Recovery by the WDT increases STOP mode standby current (I_{CC2}). This is
due to the WDT clock and divider circuitry that is now enabled and running to support this
recovery mode. See the product data sheet for actual I_{CC2} values.

All Z8 devices provide some form of dedicated Stop Mode Recovery circuitry. Two SMR
methods are implemented—a single fixed input pin or a flexible, programmable set of
inputs. The selected Z8 device product specification should be reviewed to determine the
SMR options available for use.

For devices that support SPI, the slave mode compare feature also serves as a SMR source.

In the simple case, a low level applied to input pin P27 triggers a SMR. To use this mode,
pin P27 (I/O Port 2, bit 7) must be configured as an input before STOP mode is entered.
The low level on P27 must meet a minimum pulse width T_{WSM}. (See the product data
sheet) to trigger the device reset mode). Some Z8 devices provide multiple SMR input
sources. The appropriate SMR source is selected via the SMR Register.

Use of specialized SMR modes (P2.7 input or SMR register based) or the WDT time-out
(only when in STOP mode) provide a unique reset operation. Some control registers are
initialized differently for a SMR/WDT triggered POR than a standard reset operation. See
the product specification (register file map) for exact details.

To determine the actual STOP mode current (I_{CC2}) value for the optional SMR modes
available, see the selected Z8 device's product data sheet.

STOP mode current (I_{CC2}) is minimized when:

- V_{CC} is at the low end of the devices operating range

- WDT is off in STOP mode

- Output current sourcing is minimized

- All inputs (digital and analog) are at the low or high rail voltages

Stop Mode Recovery Register

This register selects the clock divide value and determines the mode of Stop Mode Recovery (see Figure 103). All bits are Write-Only, except bit 7, that is Read-Only. Bit 7 is a flag bit that is hardware set on the condition of stop-recovery and reset by a power-on cycle. Bit 6 controls whether a low level or a high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4, of the SMR register, specify the source of the Stop Mode Recovery signal. Bits 0 and 1 control internal clock divider circuitry. The SMR is located in Bank F of the ERF at address 0Bh.

```
SMR (Fh) 0B
┌──┬──┬──┬──┬──┬──┬──┬──┐
│D7│D6│D5│D4│D3│D2│D1│D0│
└──┴──┴──┴──┴──┴──┴──┴──┘
```

SCLK ÷ TCLK Divide-by-16
0 OFF **
1 ON

External Clock Divide by 2
0 SCLK ÷ TCLK = XTAL ÷ 2*
1 SCLK ÷ TCLK = XTAL

Stop-Mode Recovery Source
000 POR Only and/or External Reset
001 P30
010 P31
011 P32
100 P33
101 P27
110 P2 NOR 0–3
111 P2 NOR 0–7

Stop Delay
0 OFF
1 ON*

Stop Recovery Level
0 Low*
1 High

Stop Flag (Read Only)
0 POR*
1 Stop Recovery
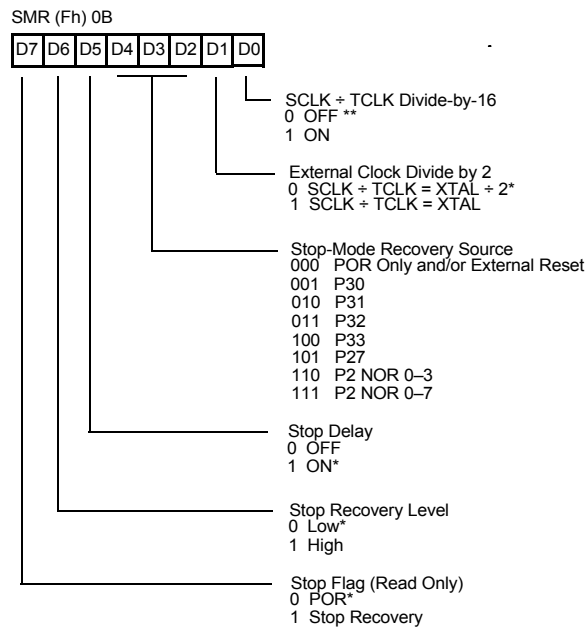
* Default setting after RESET.
** Default setting after RESET and Stop-Mode Recovery.

Figure 103. Stop-Mode Recovery Register (Write-Only Except Bit D7, Which Is Read-Only)

The SMR register is available in select Z8 CPU products. Refer to the device product specification to determine SMR options available.

SCLK ÷ TCLK Divide-by-16 Select—This D0 bit of the SMR controls a divide-by-16 prescaler of SCLK ÷ TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

External Clock Divide-by-Two—This D1 bit can eliminate the oscillator divide-by-two circuitry. When this bit is 0, the System Clock (SCLK) and Timer Clock (TCLK) are equal to the external clock frequency divided by two. The SCLK ÷ TCLK is equal to the

external clock frequency when this bit is set (D1 = 1). Using this bit together with D7 of PCON helps further lower EMI (D7 (PCON) = 0, D1 (SMR) = 1). The default setting is zero.

Stop Mode Recovery Source—The D2, D3, and D4 bits of the SMR specify the wake-up source of the stop-recovery and (Table 22 and Figure 104 on page 114).

Table 22. Stop Mode Recovery Source

SMR: 432			
D4	**D3**	**D2**	**Description of Operation**
0	0	0	POR and/or external reset recovery.
0	0	1	P30 transition.
0	1	0	P31 transition (not in Analog Mode).
0	1	1	P32 transition (not in Analog Mode).
1	0	0	P33 transition (not in Analog Mode).
1	0	1	P27 transition.
1	1	0	Logical NOR of P20 through P23.
1	1	1	Logical NOR of P20 through P27.

Stop Mode Recovery Delay Select—This D5 bit, if High, enables the T_{POR} $\overline{RESET}$ delay after Stop Mode Recovery. The default configuration of this bit is 1. If the *fast* wake up is selected, the Stop Mode Recovery source is kept active for at least 5 TpC.

Stop Mode Recovery Edge Select—A 1 in this D6 bit position indicates that a high level on any one of the recovery sources wakes the Z8® CPU from STOP mode. A 0 indicates low-level recovery. The default is 0 on POR (see Figure 104 on page 114).

Cold or Warm Start—This D7 bit is set by the device upon entering STOP mode. A 0 in this bit (cold) indicates that the device reset by POR/WDT RESET. A 1 in this bit (warm) indicates that the device awakens by a SMR source.
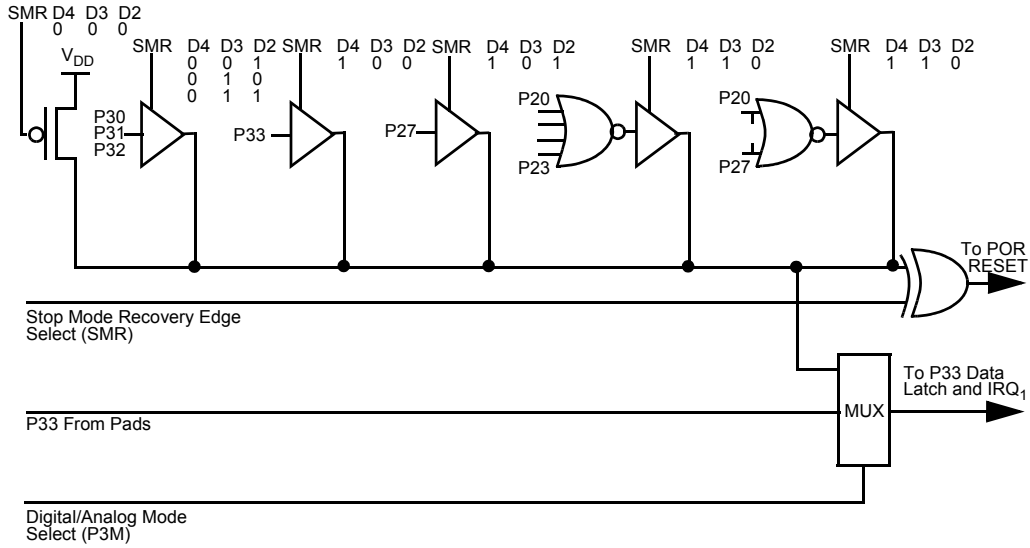
Figure 104. Stop Mode Recovery Source

If P31, P32, or P33 are to be used for a SMR source, the digital mode of operation must be selected prior to entering STOP mode.

Serial Input/Output

UART Introduction

Some Z8® CPU microcontrollers contain an on-board full-duplex Universal Asynchronous Receiver/Transmitter (UART) for data communications. The UART consists of a serial input/output (SIO) Register located at address F0h, and its associated control logic (see Figure 105). The SIO is actually two registers, the receiver buffer and the transmitter buffer, which are used in conjunction with Counter/Timer T0 and Port 3 I/O lines P30 (input) and P37 (output). Counter/Timer T0 provides the clock input for control of the data rates.
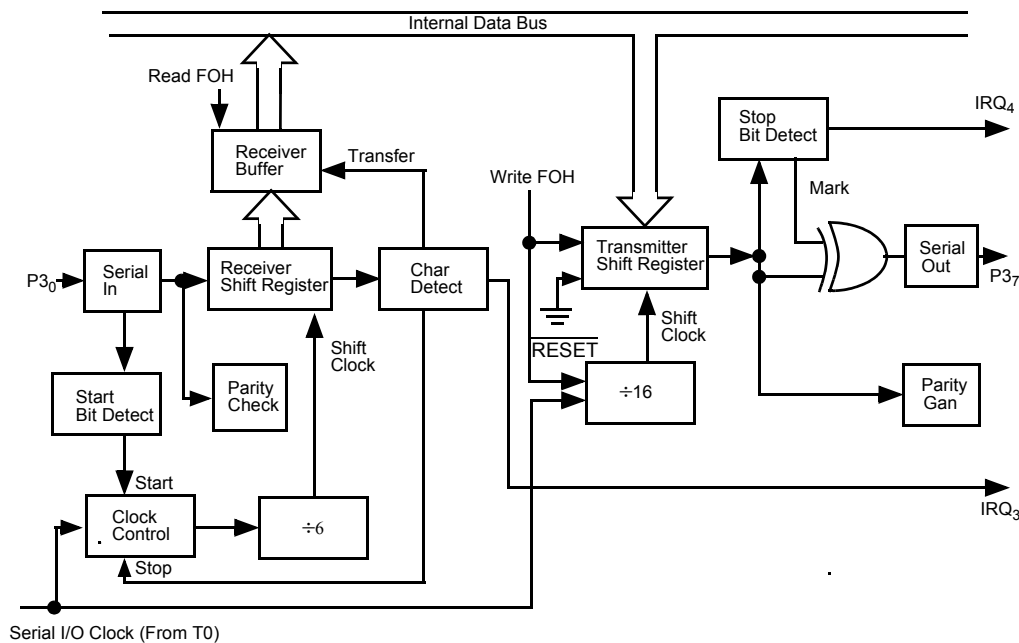
Figure 105. UART Block Diagram

Configuration of the UART is controlled by the Port 3 Mode (P3M) Register located at address F7h. Z8® CPU always transmits eight bits between the start and stop bits (eight Data Bits or seven Data Bits and one Parity Bit). Odd parity generation and detection is supported.

The SIO Register and its associated Mode Control Registers are mapped into the Standard Z8 Register File as listed in Table 23 on page 116. The organization allows the software to access the UART as general-purpose registers, eliminating the requirement for special instructions.

Table 23. UART Register Map

Register Name	Identifier	Hex Address
Port 3 Mode	P3M	F7
T0 Prescaler	PRE0	F5
Timer/Counter0	T0	F4
Timer Mode	TMR	F1
UART	SIO	F0

UART Bit-Rate Generation

When Port 3 Mode Register bit 6 is set to 1, the UART is enabled and T0 automatically becomes the bit rate generator (see Figure 106). The end-of-count signal of T0 no longer generates Interrupt Request IRQ4. Instead, the signal is used as the input to the divide-by-16 counters (one each for the receiver and the transmitter) that clock the data stream.

Register F7h
Port 3 Mode Register (P3M)
(Write-Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

0 P30 Input and P37 = Output
1 P30 Serial In and P37 = Serial Out
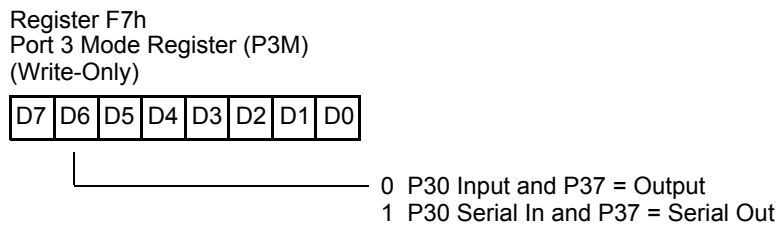
Figure 106. Port 3 Mode Register and Bit-Rate Generation

The divide chain that generates the bit rate is displayed in Figure 107. The bit rate is given by the following equation:

Bit Rate = XTAL Frequency ÷ (2 x 4 x p x t x 16)

where p and t are the initial values in Prescaler0 and Counter/Timer0, respectively. The final divide-by-16 is required because T0 runs at 16 times the bit rate in order to synchronize on the incoming data.

f_{XTAL} → ÷2 → ÷4 → P → t → ÷16 → Bit Rate Clock

PRE0   T0
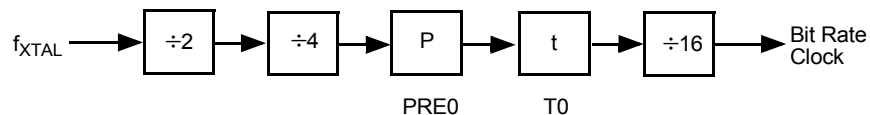
Figure 107. Bit Rate Divide Chain

To configure Z8® CPU for a specific bit rate, appropriate values as determined by the above equation must be loaded into registers PRE0

(F5h) and T0 (F4h). PRE0 also controls the counting mode for T0 and should therefore be set to CONTINUOUS mode (D0 = 1).

For example, given an input clock frequency (XTAL) of 11.9808 MHz and a selected bit rate of 1200 bits per second, the equation is satisfied by p = 39 and t = 2. Counter/Timer T0 should be set to 02h. With T0 in Continuous Mode, the value of PRE0 becomes 9Dh (see Figure 108).

Table 24 lists several commonly used bit rates and the values of XTAL, p, and t required to derive them. This list is presented for convenience and is not intended to be exhaustive.

Table 24. Bit Rates

	7,3728		7,9872		9,8304		11,0592		11,6736		11,9808		12,2880	
Bit Rate	**p**	**t**	**p**	**t**	**p**	**t**	**p**	**t**	**p**	**t**	**p**	**t**	**p**	**t**
19200	3	1	–	–	4	1	–	–	–	–	–	–	5	1
9600	3	2	–	–	4	2	9	1	–	–	–	–	5	2
4800	3	4	13	1	4	4	9	2	19	1	–	–	5	4
2400	3	8	13	2	4	8	9	4	19	2	39	1	5	8
1200	3	16	13	4	4	16	9	8	19	4	39	2	5	16
600	3	32	13	8	4	32	9	16	19	8	39	4	5	32
300	3	64	13	16	4	64	9	32	19	16	39	8	5	64
150	3	128	13	32	4	128	9	64	19	32	39	16	5	128
110	3	175	3	189	4	175	5	157	4	207	17	50	8	109

Register F5h
Prescalar 0 Register (PRE0)
(Write-Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Count Mode
$0 = T_0$ Single Pass
$1 = T_0$ Modulo-n

(Range: 1-64 decimal, 01h–00h)
(Range: 1-64)
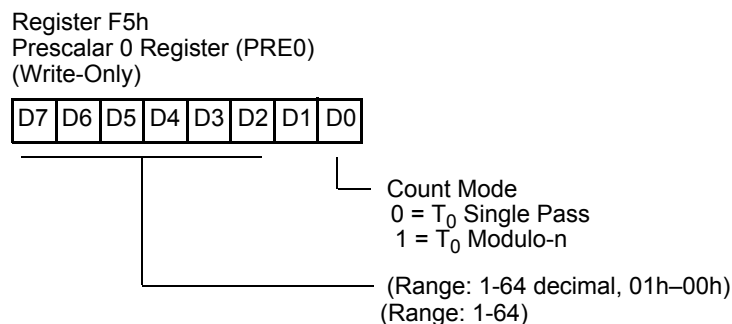
Figure 108. Prescaler 0 Register Bit-Rate Generation

The bit rate generator is started by setting the Timer Mode Register (TMR) (F1h) bit 1 and bit 0 both to 1 (see Figure 109). This transfers the contents of the Prescaler 0 Register and Counter/Timer0 Register to their corresponding down counters. In addition, counting is enabled so that UART operations begin.

Register F1h
Timer Mode Register (TMR)
(Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

0 = No Function
1 = Load T_0

0 = Disable T_0 Count
1 = Enable T_0 Count
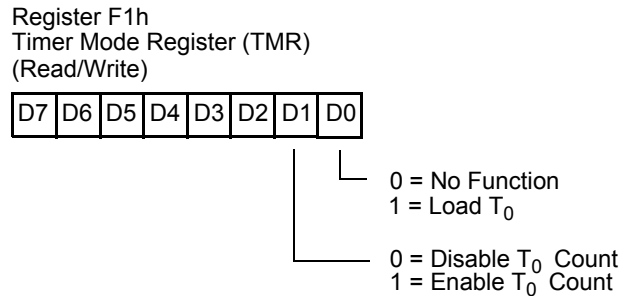
Figure 109. Timer Mode Register Bit Rate Generation

UART Receiver Operation

The receiver consists of a receiver buffer (SIO Register [F0h]), a serial-in, parallel-out shift register, parity checking, and data synchronizing logic. The receiver block diagram is displayed in Figure 105 on page 115.

Receiver Shift Register

After a hardware reset or after a character has been received, the Receiver Shift Register is initialized to all 1s and the shift clock is stopped. Serial data, input through Port 3 bit 0, is synchronized to the internal clock by two D-type flip-flops before being input to the Shift Register and the start bit detection circuitry.

The start bit detection circuitry monitors the incoming data stream, looking for a start bit (a High-to-Low input transition). When a start bit is detected, the shift clock logic is enabled. The T0 input is divided-by-16 and, when the count equals eight, the divider outputs a shift clock. This clock shifts the start bit into the Receiver Shift Register at the center of the bit time. Before the shift actually occurs, the input is rechecked to ensure that the start bit is valid. If the detected start bit is false, the receiver is reset and the process of looking for a start bit is repeated. If the start bit is valid, the data is shifted into the Shift Register every sixteen counts until a full character is assembled (see Figure 110 on page 119).
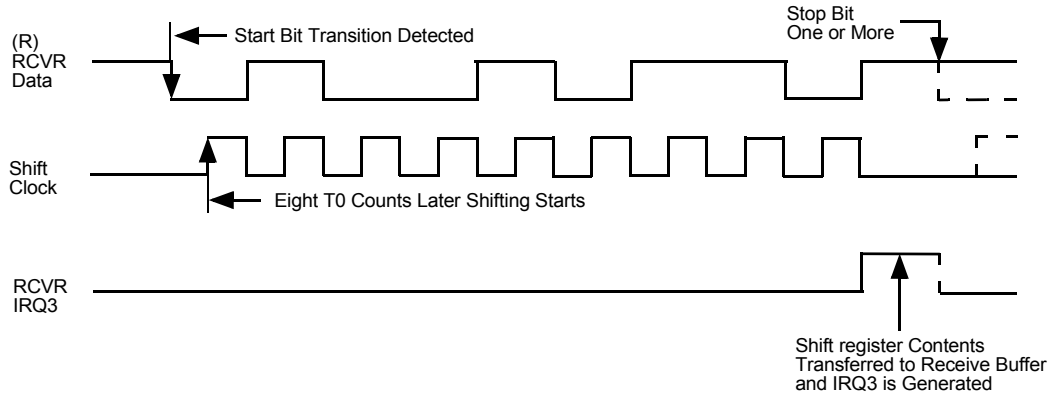
Figure 110. Receiver Timing

After a full character has been assembled in the receiver's buffer, SIO Register (F0h), Interrupt Request IRQ3 is generated. The shift clock is stopped and the Shift Register reset to all 1s. The start bit detection circuitry begins monitoring the data input for the next start bit. This cycle allows the receiver to synchronize on the center of the bit time for each incoming character.

Overwrites

Although the receiver is single buffered, it is not protected from being overwritten, so the software must read the SIO Register (F0h) within one character time after the interrupt request (IRQ3). Z8® CPU does not have a flag to indicate this overrun condition. If polling is used, the IRQ3 bit in the Interrupt Request Register must be reset by software.

Framing Errors

Framing error detection is not supported by the receiver hardware, but by responding to the interrupt request within one character bit time, the software can test for a stop bit on P30. Port 3 bits are always readable, which facilitates break detection. For example, if a null character is received, testing P30 results in a 0 being read.

Parity

The data format supported by the receiver must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the data received will be replaced by a Parity Error Flag. A parity error sets bit 7 to 1, otherwise, bit D7 is set to 0. Figure 111 on page 120 displays these data formats.
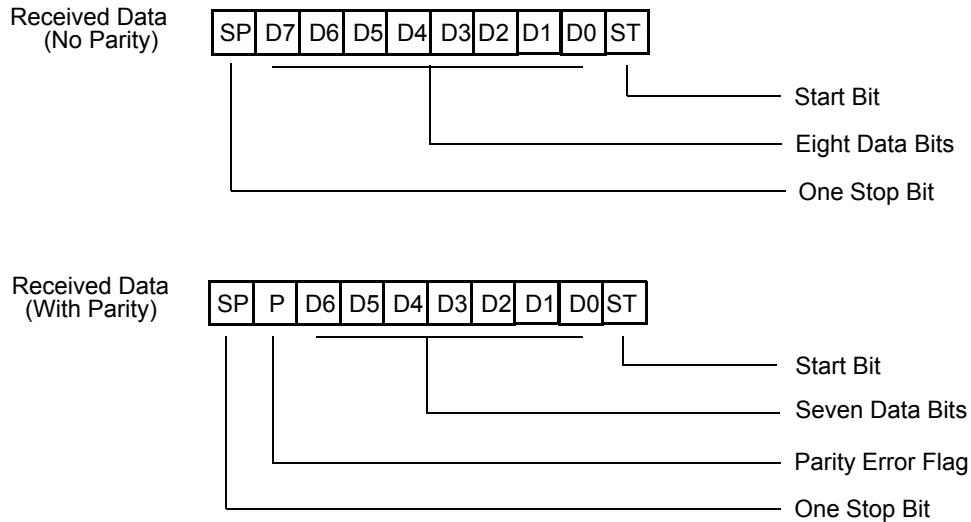
Figure 111. Receiver Data Formats

Z8® CPU hardware supports odd parity only, that is enabled by setting the Port 3 Mode Register bit 7 to 1 (see Figure 112). If even parity is required, PARITY mode should be disabled (P3M bit 7 set to 0), and software must calculate the received data's parity.
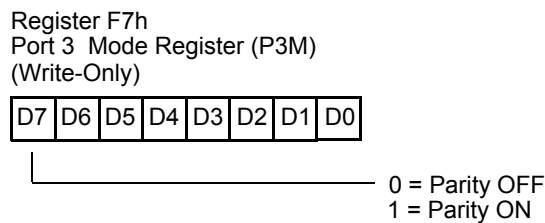
Figure 112. Port 3 Mode Register Parity

Transmitter Operation

The transmitter consists of a transmitter buffer (SIO Register [F0h]), a parity generator, and associated control logic. The transmitter block diagram is displayed as part of Figure 105 on page 115.

After a hardware reset or after a character has been transmitted, the transmitter is forced to a marking state (output always High) until a character is loaded into the transmitter buffer, SIO Register (F0h). The transmitter is loaded by specifying the SIO Register as the destination register of any instruction.

T0's output drives a divide-by-16 counter that in turn generates a shift clock every 16 counts. This counter is reset when the transmitter buffer is written by an instruction. This reset synchronizes the shift clock to the software. The transmitter then outputs one bit per shift clock, through Port 3 bit 7, until a start bit, the character written to the buffer, and two stop bits have been transmitted. After the second stop bit has been transmitted, the output is again forced to a marking state. Interrupt request IRQ4 is generated at this time to notify the processor that the transmitter is ready to accept another character.

Overwrites

The user is not protected from overwriting the transmitter, so it is up to the software to respond to IRQ4 appropriately. If polling is used, the IRQ4 bit in the Interrupt Request Register must be reset.

Parity

The data format supported by the transmitter has a start bit, eight data bits, and at least two stop bits. If parity is on, bit 7 of the data transmitted will be replaced by an odd parity bit. Figure 113 displays the transmitter data formats.

Parity is enabled by setting Port 3 Mode Register bit 7 to 1. If even parity is required, PARITY mode should be disabled (P3M bit 7 reset to 0), and software must modify the data to include even parity.

Because the transmitter can be overwritten, you can generate a break signal. This is done by writing null characters to the transmitter buffer (SIO Register [F0h]) at a rate that does not allow the stop bits to be output. Each time the SIO Register is loaded, the divide-by-16 counter is resynchronized and a new start bit is output followed by data.
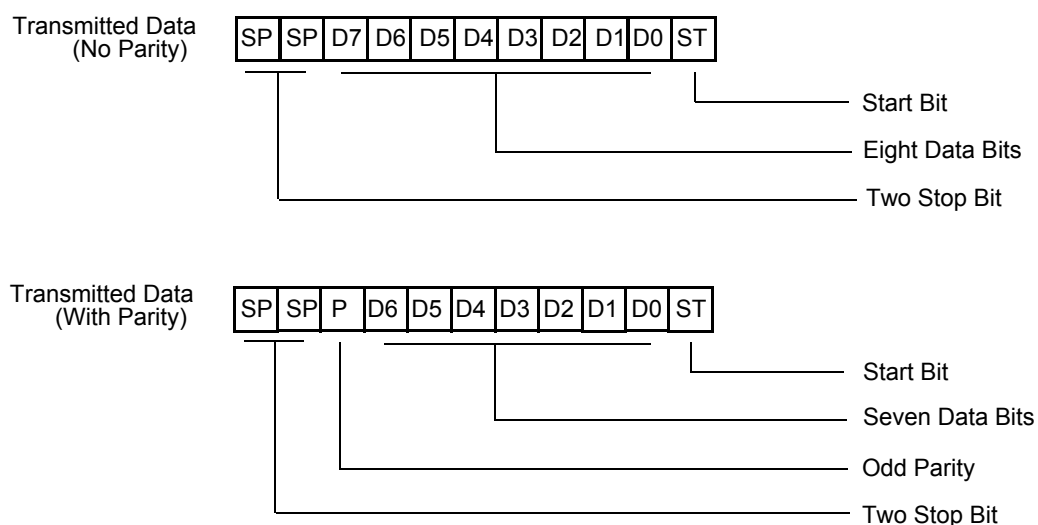
Figure 113. Transmitter Data Formats

UART Reset Conditions

After a hardware reset, the SIO Register contents are undefined, and Serial Mode and parity are disabled. Figure 114 and Figure 115 display the binary reset values of the SIO Register and its associated mode register P3M.

Register RF0h
Serial I/O Register (SIO)
(Read/Write)

| U | U | U | U | U | U | U | U |

Serial Data (D_0 = LSB)
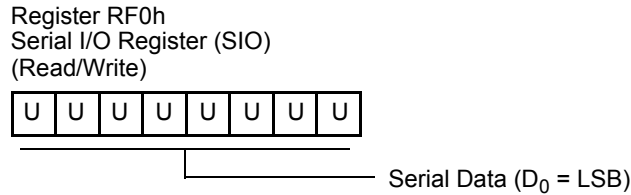
Figure 114. SIO Register Reset

Register F7h
Port 3 Mode Register (P3M)
(Write-Only)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0 Port 2 Pull-Ups Open-Drain
1 Port 2 Pull-Ups Active

0 P32 = Input   P35 = Output
1 P32 = DAV0/RDY0   P35 = RDY0/DAV0

00 P33 = Input   P34 = Output
01 P33 = Input   P34 = DM
10 P33 = Input   P34 = DM
11 P33 = DAV1/RDY1   P34 = RDY1/DAV1

0 P31 = Input (T_{IN})   P36 = Output (T_{OUT})
1 P32 = DAV2/RDY2   P36 = RDY2/DAV2

0 P30 = Input   P37 = Output
1 P30 = Serial In   P37 = Serial Out

0 Parity ON
1 Parity OFF

Figure 115. P3M Register Reset

Serial Peripheral Interface

Select Z8 microcontrollers incorporate a Serial Peripheral Interface (SPI) for communication with other microcontrollers and peripherals. The SPI includes features such as Stop-

Mode Recovery, Master/Slave selection, and Compare mode. Table 25 contains the pin configuration for the SPI feature when it is enabled. The SPI consists of four registers: SPI Control Register (SCON), SPI Compare Register (SCOMP), SPI Receive/Buffer Register (RxBUF), and SPI Shift Register. SCON is located in bank (C) of the Expanded Register File at address 02.

Table 25. SPI Pin Configuration

Name	Function	Pin Location
DI	Data-In	P20
DO	Data-Out	P27
SS	Slave Select	P35
SK	SPI Clock	P34

The SPI Control Register (SCON; see Figure 116 on page 124), is a read/write register that controls Master/Slave selection, interrupts, clock source and phase selection, and error flag. Bit 0 enables/disables the SPI with the default being SPI disabled. A 1 in this location enables the SPI, and a 0 disables the SPI. Bits 1 and 2 of the SCON register in Master Mode select the clock rate. You can choose whether internal clock is divide-by-2, 4, 8, or 16. In Slave Mode, Bit 1 of this register flags the user if an overrun of the RxBUF Register has occurred. The RxCharOverrun flag is only reset by writing a 0 to this bit. In slave mode, bit 2 of the Control Register disables the data-out I/O function. If a 1 is written to this bit, the data-out pin is released to its original port configuration. If a 0 is written to this bit, the SPI shifts out one bit for each bit received. Bit 3 of the SCON Register enables the compare feature of the SPI, with the default being disabled. When the compare feature is enabled, a comparison of the value in the SCOMP Register is made with the value in the RxBUF Register. Bit 4 signals that a receive character is available in the RxBUF Register.

SCON (C) 02

D7	D6	D5	D4	D3	D2	D1	D0

SPI Enable
 0 Disable *
 1 Enable

RxCharOverrun (S)
 0 Reset
 1 Overrun

CLK Divide (M)
 00 TCLK/2
 01 TCLK/4
 10 TCLK/8
 11 TCLK/16

DO SPI Port Enable (S)
 0 SPI DO Port Enable
 1 Do Port to I/O

Compare Enable
 0 Enable
 1 Disable *

RxCharAvail
 0 Reset
 1 Char. Avail

Clock Phase
 0 Trans/Fall
 1 Trans/Rise

CLK Source
 0 TCLK
 1 Timer 0 Output

Master Slave
 0 Slave
 1 Master

(S) Used with Bit D7 equal to 0
(M) Used with Bit D7 equal to 1

* Default setting after Reset

Figure 116. SPI Control Register

If the associated IRQ3 is enabled, an interrupt is generated. Bit 5 controls the clock phase of the SPI. A 1 in bit 5 allows for receiving data on the clock's falling edge and transmitting data on the clock's rising edge. A 0 allows receiving data on the clock's rising edge and transmitting on the clock's falling edge. The SPI clock source is defined in bit 6. A 1 uses Timer0 output for the SPI clock, and a 0 uses TCLK for clocking the SPI. Finally, bit 7 determines whether the SPI is used as a Master or a Slave. A 1 puts the SPI into Master mode and a 0 puts the SPI into Slave mode.

SPI Operation

The SPI is used in one of two modes: either as system slave, or as system master. Several of the possible system configurations are displayed in Figure 117 on page 126. In slave mode, data transfer starts when the slave select (SS) pin goes active. Data is transferred into the slave's SPI Shift Register through the DI pin, which has the same address as the RxBUF Register. After a byte of data has been received by the SPI Shift Register, a Receive Character Available (RCA/IRQ3) flag and interrupt is generated. The next byte of data is received at this time. The RxBUF Register must be cleared, or a Receive Character Overrun (RxCharOverrun) flag is set in the SCON Register, and the data in the RxBUF Register is overwritten. When the communication between the master and slave is complete, the SS goes inactive.When the SPI is activated as a slave, it operates in all system modes: STOP, HALT, and RUN.

Unless disconnected, for every bit that is transferred into the slave through the DI pin, a bit is transferred out through the D0 pin on the opposite clock edge. During slave operation, the SPI clock pin (SK) is an input. In master mode, the CPU must first activate a SS through one of its I/O ports. Next, data is transferred through the master's D0 pin one bit per master clock cycle. Loading data into the shift register initiates the transfer. In master mode, the master's clock drives the slave's clock. At the conclusion of a transfer, a Receive Character Available (RCA/IRQ3) flag and interrupt is generated. Before data is transferred via the D0 pin, the SPI Enable bit in the SCON Register must be enabled.

SPI Compare

When the SPI Compare Enable bit, D3 of the SCON Register is set to 1, the SPI Compare feature is enabled. The compare feature is only valid for slave mode. A compare transaction begins when the (SS) line goes active. Data is received as if it were a normal transaction, but there is no data transmitted to avoid bus contention with other slave devices. When the compare byte is received, IRQ3 is not generated. Instead, the data is compared with the contents of the SCOMP Register. If the data does not match, DO remains inactive and the slave ignores all data until the (SS) signal is reset. If the data received matches the data in the SCOMP register, then a SMR signal is generated. DO is activated if it is not tri-stated by D2 in the SCON Register, and data is received the same as any other SPI slave transaction.

Slaves' not comparing remain in their current mode, whereas slaves' comparing wake from a STOP mode by means of an SMR

SPI Clock

The SPI clock may be driven by three sources: Timer0, a division of the internal system clock, or the external master when in slave mode. Bit D6 of the SCON Register controls what source drives the SPI clock. A 0 in bit D6 of the SCON Register determines the divi-

sion of the internal system clock if this is used as the SPI clock source. Divide by 2, 4, 8, or 16 is chosen as the scaler.
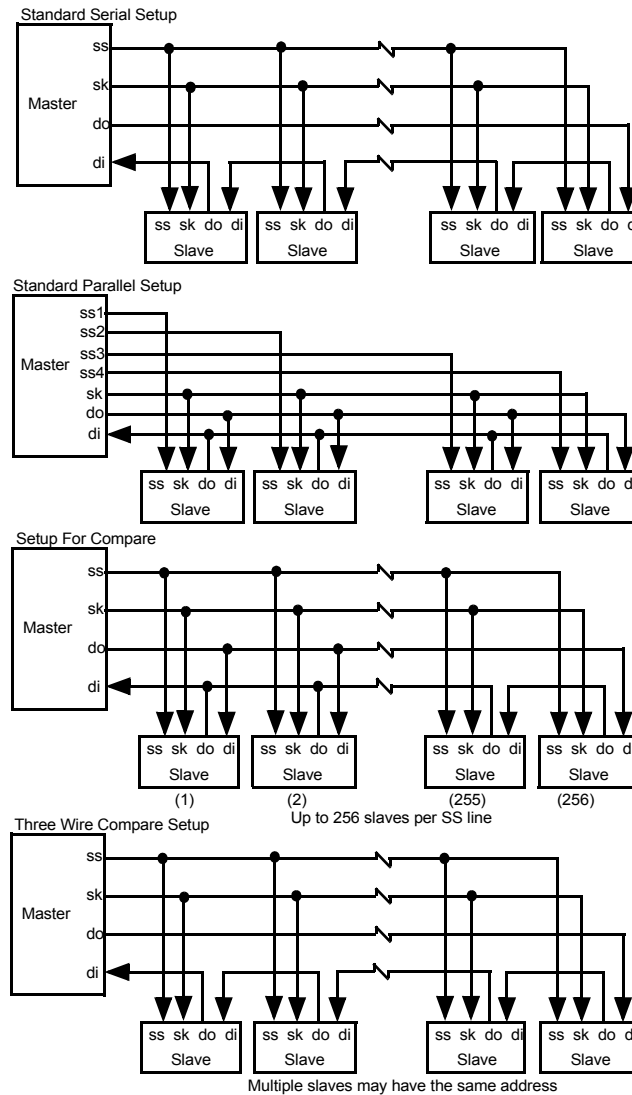
Figure 117. SPI System Configuration

Receive Character Available and Overrun

When a complete data stream is received, an interrupt is generated and the RxCharAvail bit in the SCON Register is set. Bit 4 in the SCON Register is for enabling or disabling the RxCharAvail interrupt. The RxCharAvail bit is available for interrupt polling purposes and is reset when the RxBUF Register is read. RxCharAvail is generated in both master and

slave modes. While in slave mode, if the RxBUF is not read before the next data stream is received and loaded into the RxBUF Register, Receive Character Overrun (RxCharOverrun) occurs. Because there is no requirement for clock control in slave mode, bit D1 in the SPI Control Register is used to log any RxCharOverrun (see Figure 118 and Figure 119 on page 128).
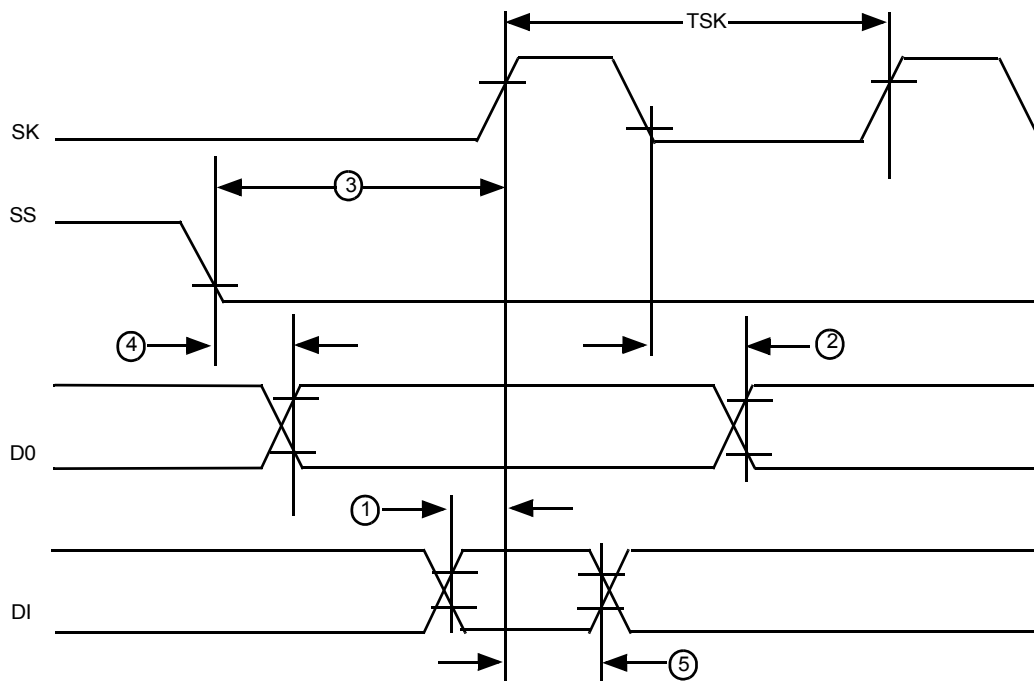
Figure 118. SPI Timing
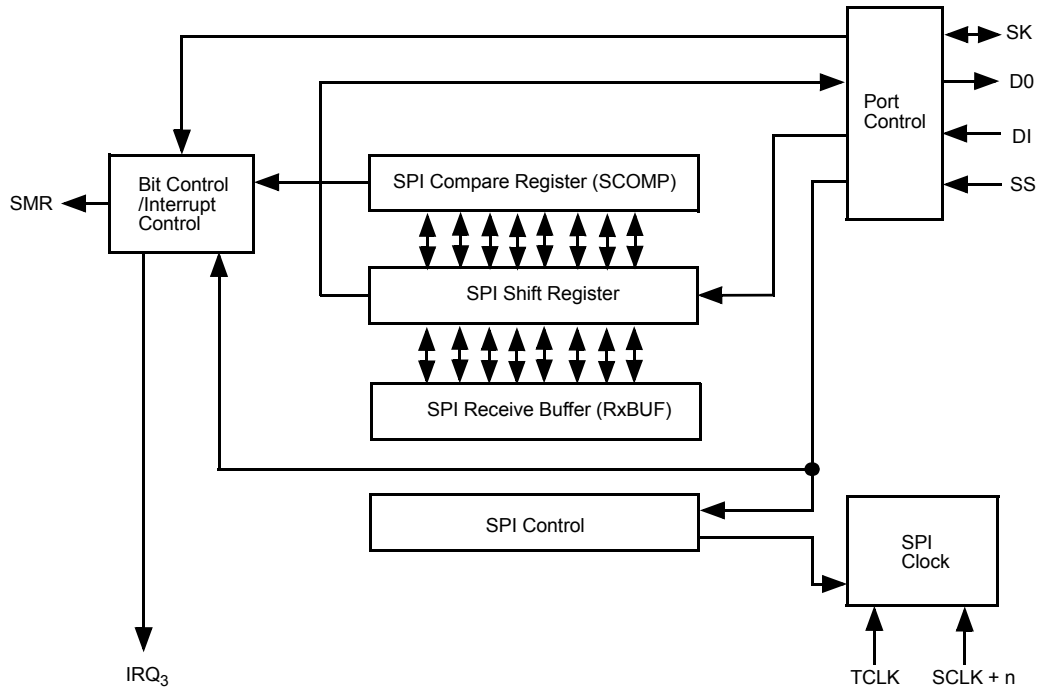
Figure 119. SPI Logic
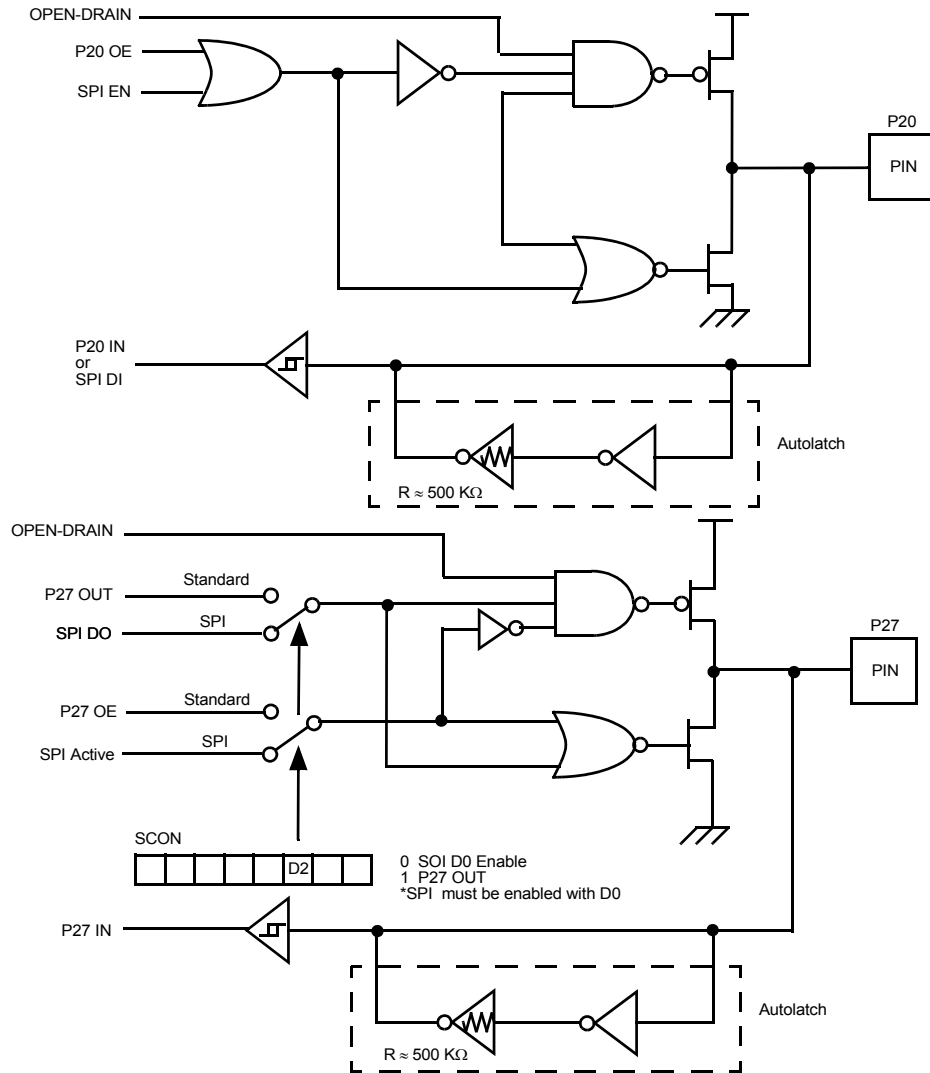
Figure 120. SPI Data In/Out Configuration
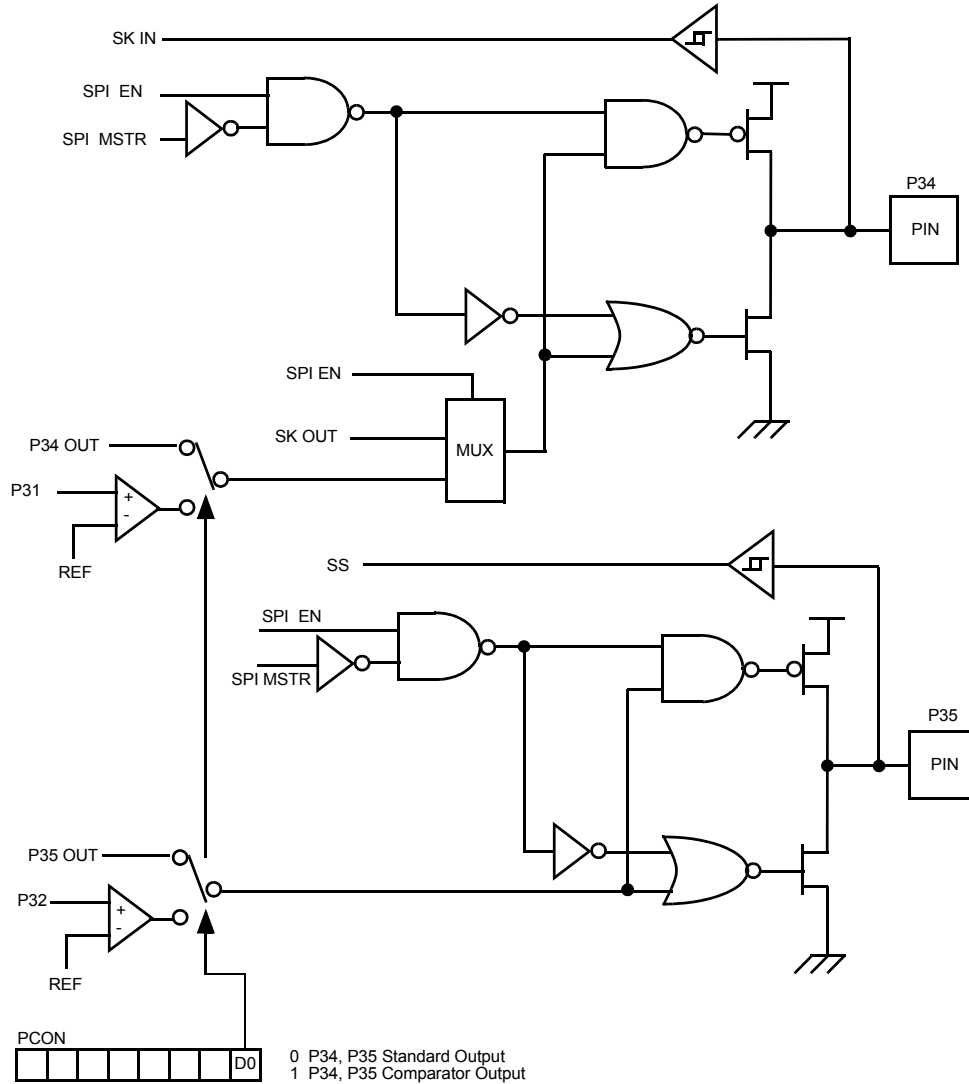
Figure 121. SPI Clock/SPI Slave Select Output Configuration

External Interface

Introduction

Z8® CPU can be a microcontroller with 20 pins for external memory interfacing. The external memory interface on the Z8 CPU is generally for either RAM or ROM; this features is only available for devices featuring Port 0, Port 1, R/$\overline{W}$, $\overline{DM}$, $\overline{AS}$, and $\overline{DS}$. Refer to specific product specifications for availability of these features.

Z8 CPU offers a multiplexed external memory interface. In multiplexed mode, eight pins from Port 1 form an Address/Data Bus (AD7–AD0), eight pins from Port 0 form a High Address Bus (A15–A8). Three additional pins provide the Address Strobe, Data Strobe, and the Read/Write Signal. Figure 122 displays the Z8 CPU external interface pins.
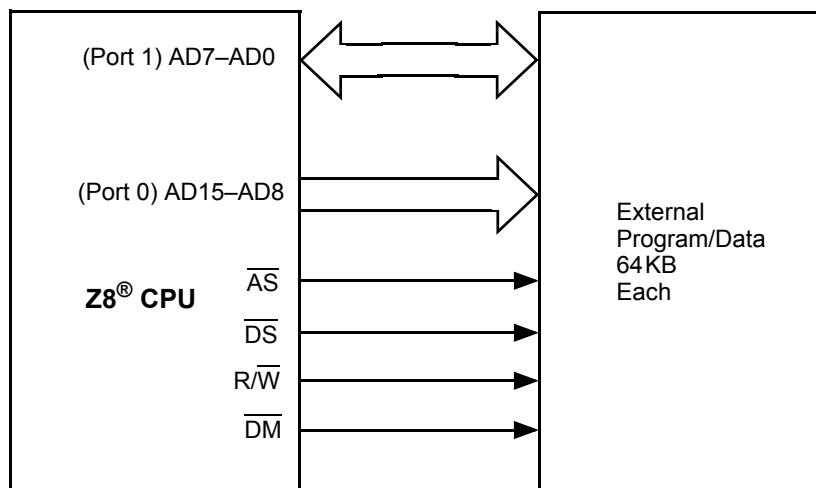
Figure 122. Z8 CPU External Interface Pins

Pin Descriptions

The following sections briefly describe the pins associated with the Z8 CPU external memory interface.

Address Strobe (output, active Low)—Address Strobe (AS) is pulsed Low once at the beginning of each machine cycle. The rising edge of $\overline{AS}$ indicates the address, Read/Write (R/$\overline{W}$), and data memory ($\overline{DM}$) signals are valid for program or data memory transfers. In some cases, the Z8 CPU address strobe is pulsed low regardless of accessing external or internal memory. Refer to specific product specifications for $\overline{AS}$ operation.

Data Strobe (Output, Active Low)—Data Strobe (DS) provides the timing for data movement to or from the Address/Data bus for each external memory transfer. During a Write Cycle, data out is valid at the leading edge of the $\overline{DS}$. During a Read Cycle, data in must be valid prior to the trailing edge of the $\overline{DS}$.

Read/Write (Output)—Read/Write (R/W) determines the direction of data transfer for memory transactions. $R/\overline{W}$ is Low when writing to program or data memory, and High for all other transactions.

Data Memory (Output)—Data memory (DM) provides a signal to separate external Program Memory from external data memory. It is a programmable function on pin P34. Data memory is active low for external data memory accesses and high for external Program Memory accesses.

High Address Lines A15–A8—A15–A8 provide the High Address lines for the memory interface. The Port 0–1 mode registers must have bits D7 and D1 set equal to 1 to configure Port 0 as A15–A8. Outputs can be CMOS- or TTL-compatible. Refer to product specifications for actual type. See Figure 123 on page 133.

Address/Data Lines AD7–AD0—AD7–AD0 is a multiplexed Address/Data memory interface. The lower eight Address lines (A7–A0) are multiplexed with Data lines (D7–D0). Port 0–1 mode registers must have bits D4 set equal to 1 and D3 set equal to 0 to configure Port 1 as AD7–AD0. Inputs and outputs are TTL-compatible. See Figure 123 on page 133.

Reset—$\overline{RESET}$ (input, active Low) initializes the Z8® CPU. When $\overline{RESET}$ is deactivated, program execution begins from program location `000Ch`. If held Low, $\overline{RESET}$ acts as a register file protect during power-down and power-up sequences. To avoid asynchronous and noisy reset problems, the Z8 CPU is equipped with a reset filter of four external clocks ($4T_PC$). If the external $\overline{RESET}$ signal is less than $4T_PC$ in duration, no reset occurs. On the fifth clock after the $\overline{RESET}$ is detected, an internal reset signal is latched and held for an internal register count of 18 or more external clocks, or for the duration of the external $\overline{RESET}$, whichever is longer. Refer to specific product specifications for length of reset delay time.

Crystal1, Crystal2 (Oscillator Input and Output)—These pins connect a parallel-resonant crystal, ceramic resonator, LC, RC network, or external single-phase clock to the on-chip oscillator input. Refer to the device product specifications for information on availability of RC oscillator features.

External Addressing Configuration

The minimum bus configuration uses Port 1 as a multiplexed address/data port (AD7–AD0), allowing access to 256 bytes of external memory. In this configuration, the eight low-order bits (A0–A7) are multiplexed with the data (D7–D0).

Port 0 can be programmed to provide either four additional address lines (A11–A8), which increases the addressable memory to 4KB, or eight additional address lines (A15–A8), which increases the addressable external memory up to 64KB. It is required to add a NOP after configuring Port 0/Port 1 for external addressing before jumping to external memory execution.

Register F8h (P01M)
Port 0–1 Mode Register (P01M)
(Write-Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

P00–P07 Mode
00 = Output
01 = Input
1X = A8–A11

P10–P17 Mode
00 = Byte Output
01 = Byte Output
10 = AD0-AD7
11 = High Impedance AD0–AD7,
     A8–A15, $\overline{AS}$, $\overline{DS}$, R/W

P04–P07 Mode
00 = Output
01 = Input
1X = A12–A15
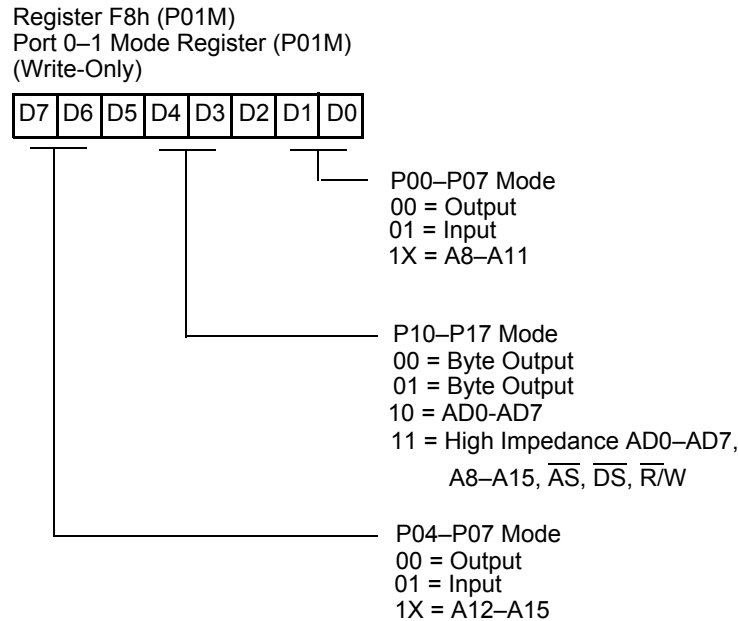
Figure 123. External Address Configuration

External Stacks

Z8® CPU architecture supports stack operations in either the Z8 Standard Register File or external data memory. A stack's location is determined by bit 2 in the Port 0–1 Mode Register (F8h). If bit 2 is set to 0, the stack is in external data memory (see Figure 124 on page 134).

The instruction used to change the stack selection bit should not be immediately followed by the instructions RET or IRET, because this causes indeterminate program flow. After a $\overline{RESET}$, the internal stack is selected.

> **Note:** *If Port 0 is configured as A15–A8 and the stack is selected as internal, any stack operation causes the contents in register* FEh *to be displayed on Port 0.*

Register F8h (P01M)
Port 0–1 Register
(Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

Z8 Stack Selection
0 = External
1 = Internal
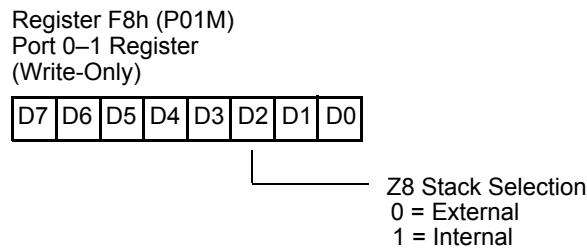
Figure 124. Z8® Stack Selection

Data Memory

The two Z8 external memory spaces, data and program, are addressed as two separate spaces of up to 64 KB each. External Program Memory and external data memory are logically selected by the data memory select output ($\overline{DM}$). $\overline{DM}$ is made available on Port 3, bit 4 (P34) by setting bit 4 and bit 3 in the Port 3 Mode Register (F7h) to 10 or 01 (see Figure 125). $\overline{DM}$ is active Low during the execution of the LDE, LDEI instructions, and High for the execution of program instructions. $\overline{DM}$ is also active Low during the execution of CALL, POP, PUSH, RET and IRET instructions if the stack resides in external data memory. After a $\overline{RESET}$, $\overline{DM}$ is not selected.

Register F7h (P3M)
Port 3 Mode Register
(Write-Only)

D7	D6	D5	D4	D3	D2	D1	D0

Bits	Configuration	
00	P33 = Input	P34 = Output
01	P33 = Input	P34 = $\overline{DM}$
10	P33 = Input	P34 = DM
11	P33 = DAV1/RDY1	P34 = RDY1/$\overline{DAV1}$
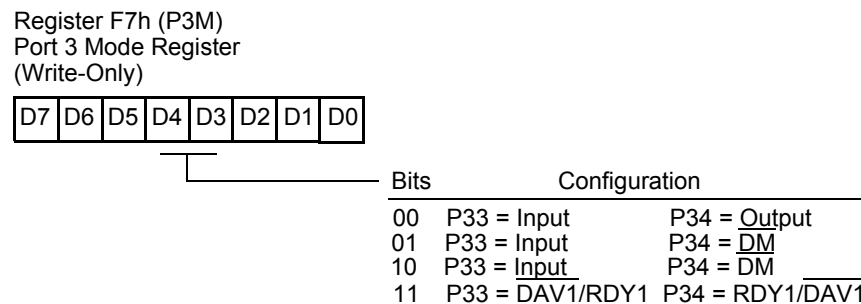
Figure 125. Port 3 Data Memory Operation

Bus Operation

Typical data transfers between Z8® CPU and external memory are displayed in Figure 126 on page 135 and Figure 127 on page 136. Machine cycles can vary from 6 to 12 clock periods depending on the operation being performed. The notations used to describe the basic timing periods of the Z8 CPU are machine cycles (Mn), timing states (Tn), and clock periods. All timing references are made with respect to the output signals $\overline{AS}$ and $\overline{DS}$. The

clock is shown for clarity only and does not have a specific timing relationship with other signals.
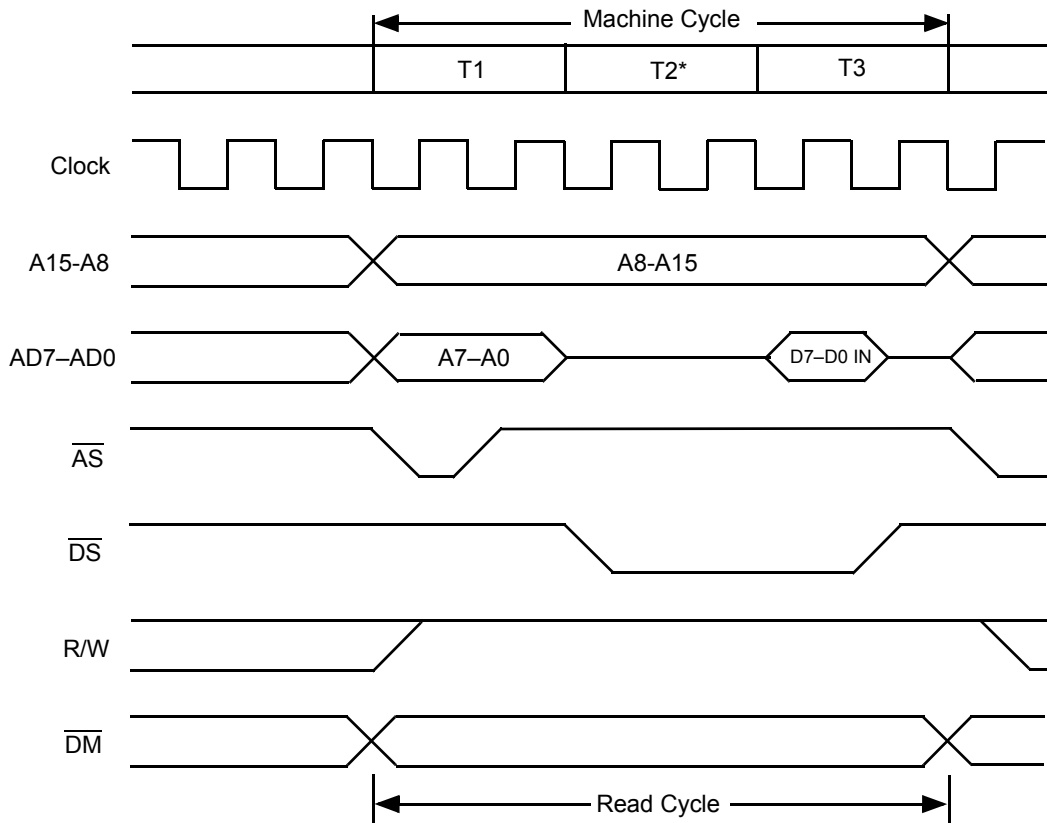
*Port inputs are strobed during T2, which is two internal systems clocks before the execution cycle of the current instruction.

Figure 126. External Instruction Fetch or Memory Read Cycle
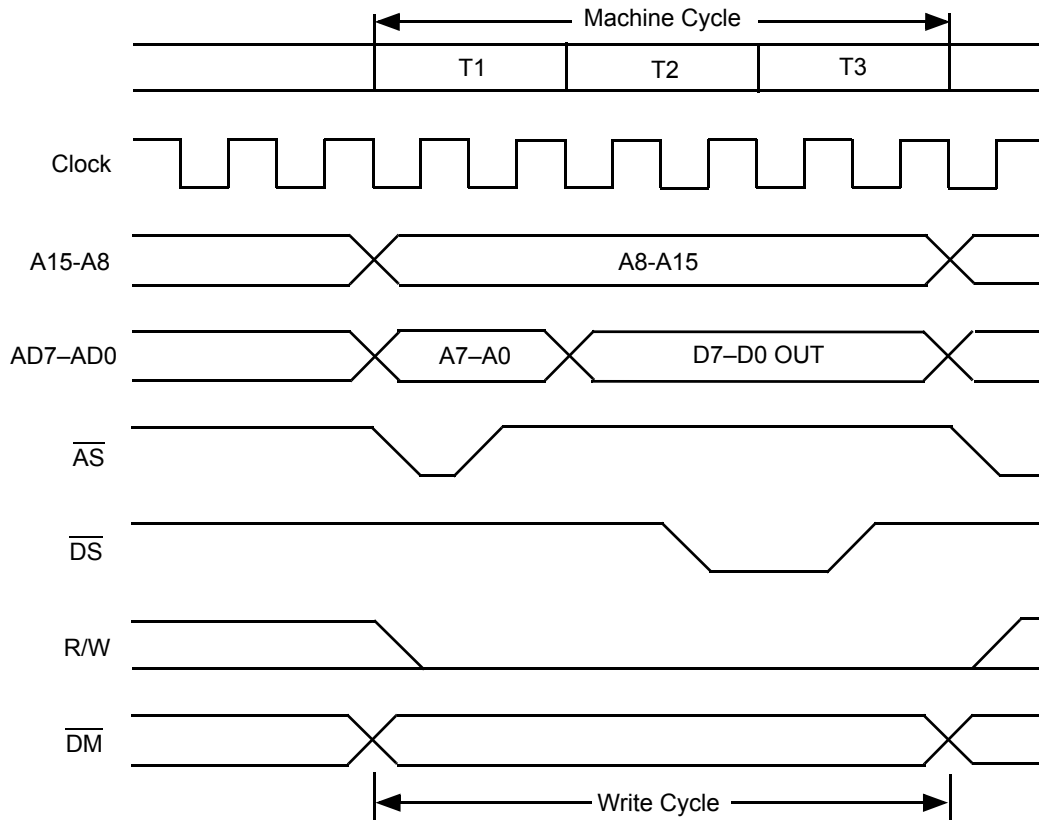
Figure 127. External Memory Write Cycle

Address Strobe

All transactions start with $\overline{\text{AS}}$ driven Low and then raised High by Z8® CPU. The rising edge of $\overline{\text{AS}}$ indicates that R/$\overline{\text{W}}$, $\overline{\text{DM}}$ (if used), and the address outputs are valid. The address outputs (AD7–AD0), remain valid only during MnT1 and typically must be latched using $\overline{\text{AS}}$. Address outputs (A15–A8) remain stable throughout the machine cycle, regardless of the addressing mode.

Data Strobe

Z8® CPU uses $\overline{\text{DS}}$ to time the actual data transfer. For Write operations (R/$\overline{\text{W}}$ = Low), a Low on $\overline{\text{DS}}$ indicates that valid data is on the AD7–AD0 lines. For Read operations (R/W = High), the bus is placed in a high-impedance state before driving $\overline{\text{DS}}$ Low, so the addressed device can put its data on the bus. The Z8 CPU samples this data prior to raising $\overline{\text{DS}}$ High.

Extended Bus Timing

Some products can accommodate slow memory access time by automatically inserting an additional software controlled state time (Tx). This stretches the $\overline{DS}$ timing by two clock periods. Figure 128 and Figure 129 on page 138 display extended external memory Read and Write cycles.
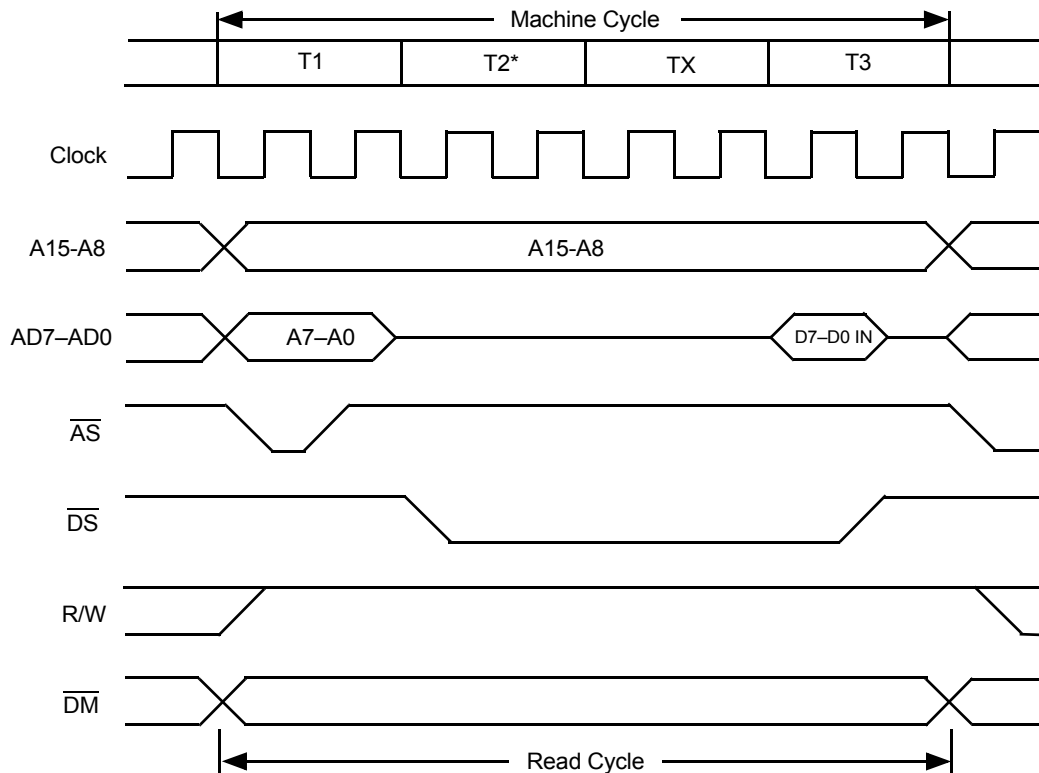
*Port inputs are strobed during T2, which is two internal system clocks
 before the execution of the current instruction.

Figure 128. Extended External Instruction Fetch or Memory Read Cycle
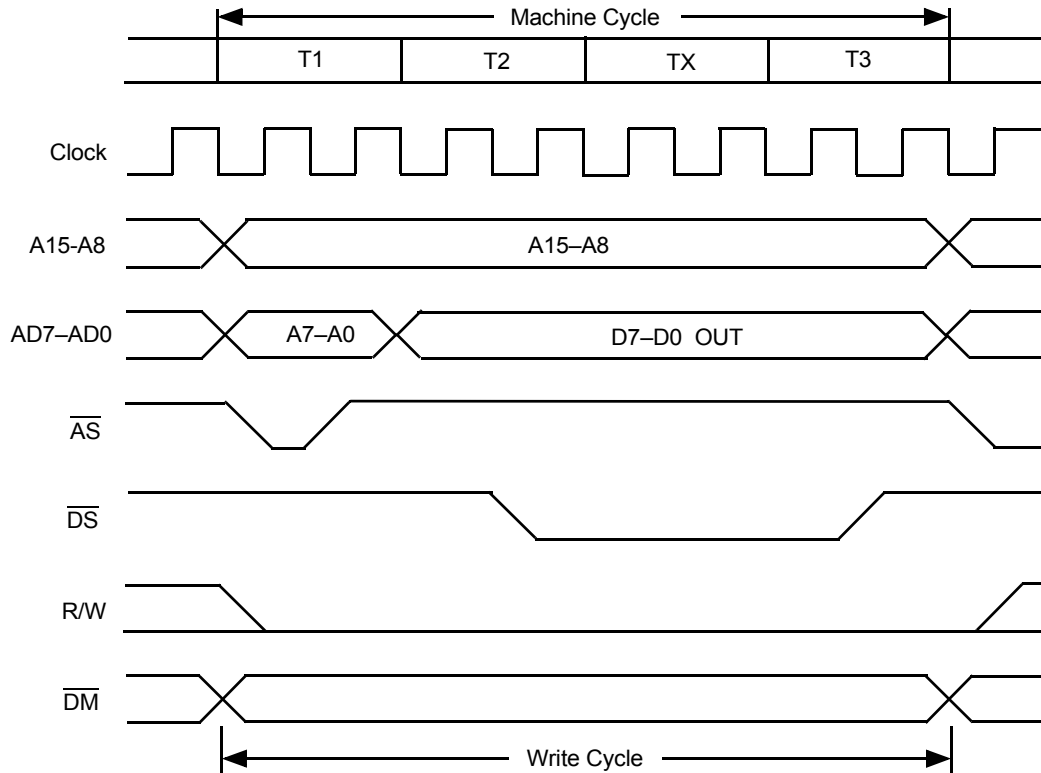
Figure 129. Extended External Memory Write Cycle

Timing is extended by setting bit D5 in the Port 0–1 Mode Register (F8h) to 1 (see Figure 130). After a RESET, this bit is set to 0.
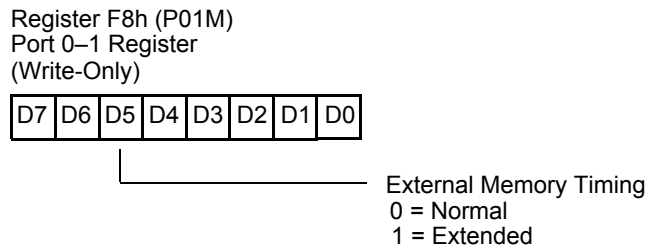
Figure 130. Extended Bus Timing

Instruction Timing

The High throughput of Z8® CPU is due, in part, to the use of an instruction pipeline, in which the instruction fetch and execution cycles are overlapped. During the execution of

the current instruction, the opcode of the next instruction is fetched. Instruction pipelining is displayed in Figure 131.

Figure 131 and Figure 132 on page 139 display typical instruction cycle timing for instructions fetched from memory. For those instructions that require execution time longer than that of the overlapped fetch, or reference program or data memory as part of their execution, the pipe must be flushed.Figure 131 and Figure 132 on page 139 assume the XTAL ÷ 2 clock mode is selected.
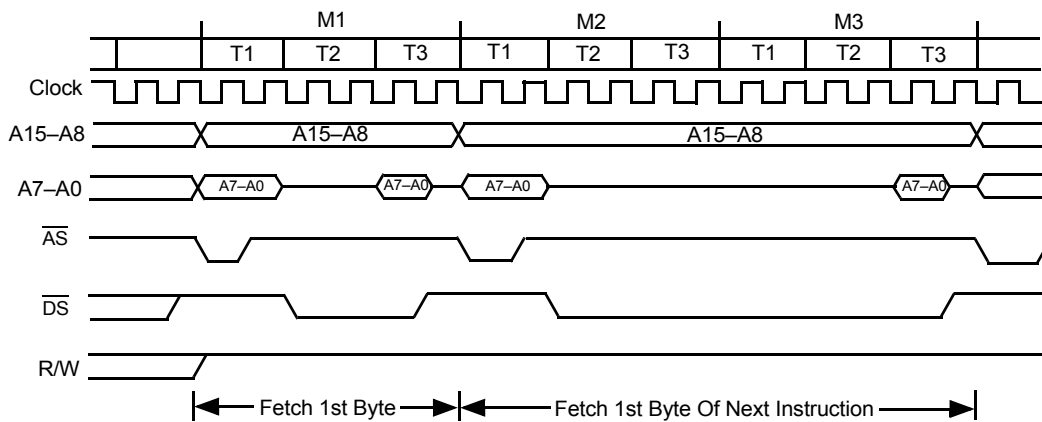
* Port inputs are strobed during T2, which is two internal system clocks
before the execution cycle of the current installation

Figure 131. Instruction Cycle Timing (1-Byte Instructions)
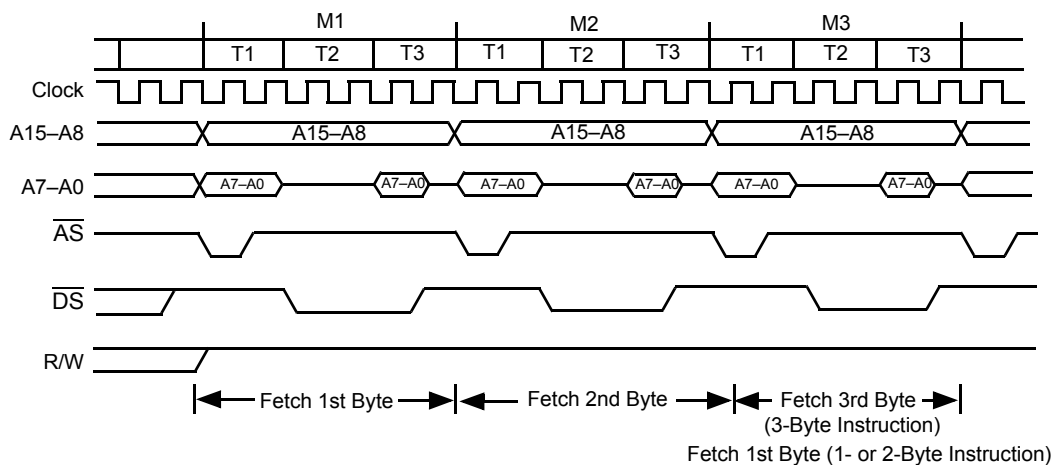
*Port inputs are strobed during T2, which is two internal system clocks before
the execution cycle of the current instruction.

Figure 132. Instruction Cycle Timing (2- and 3-Byte Instructions)

Z8® Reset Conditions

After a hardware reset, extended timing is set to accommodate slow memory access during the configuration routine, $\overline{DM}$ is inactive, the stack resides in the register file. Port 0, 1, and 2 are reset to input mode. Port 2 is set to Open-Drain Mode.

Instruction Set

Z8® instructions can be divided functionally into the following eight groups:

- Load

- Bit Manipulation

- Arithmetic

- Block Transfer

- Logical

- Rotate and Shift

- Program Control

- CPU Control

The following summary shows the instructions belonging to each group and the number of operands required for each. The source operand is `src`, the destination operand is `dst`, and a condition code is `cc`.

Table 26. Load Instructions

Mnemonic	Operands	Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant
LDE	dst, src	Load External
POP	dst	Pop
PUSH	src	Push

Table 27. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADD	dst, src	Add
CP	dst, src	Compare
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word

Table 27. Arithmetic Instructions (Continued)

Mnemonic	Operands	Instruction
INC	dst	Increment
INCW	dst	Increment Word
SBC	dst, src	Subtract with Carry
SUB	dst, src	Subtract

Table 28. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical Exclusive OR

Table 29. Program Control Instructions

Mnemonic	Operands	Instruction
CALL	dst	Call Procedure
DJNZ	dst, src	Decrement and Jump Non-Zero
IRET		Interrupt Return
JP	cc, dst	Jump
JR	cc, dst	Jump Relative
RET		Return

Table 30. Bit Manipulation Instructions

Mnemonic	Operands	Instruction
TCM	dst, src	Test Complement Under Mask
TM	dst, src	Test Under Mask
AND	dst, src	Bit Clear
OR	dst, src	Bit Set
XOR	dst, src	Bit Complement

Table 31. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant Auto Increment
LDEI	dst, src	Load External Auto Increment

Table 32. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
RL	dst	Rotate Left
RLC	dst	Rotate Left Through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right Through Carry
SRA	dst	Shift Right Arithmetic
SWAP	dst	Swap Nibbles

Table 33. CPU Control Instructions

Mnemonic	Operands	Instruction
CCF		Complement Carry Flag
DI		Disable Interrupts
EI		Enable Interrupts
HALT		Halt
NOP		No Operation
RCF		Reset Carry Flag
SCF		Set Carry Flag
SRP	src	Set Register Pointer
STOP		Stop
WDh		WDT Enable During HALT
WDT		WDT Enable or Refresh

Processor Flags

The Flag Register (FCh) informs about the current status of Z8® CPU. The flags and their bit positions in the Flag Register are displayed in Figure 133.

Z8 Flag Register contains six bits of status information which are set or cleared by CPU operations. Four of the bits (C, V, Z, and S) can be tested for use with conditional Jump instructions. Two flags (H and D) cannot be tested and are used for BCD arithmetic. The two remaining bits in the Flag Register (F1 and F2) are available, but they must be set or cleared by instructions and are not usable with conditional Jumps.

As with bits in the other control registers, the Flag Register bits can be set or reset by instructions; however, only those instructions that do not affect the flags as an outcome of the execution should be used (Load Immediate).

▶ **Note:** *The Watchdog Timer (WDT) instruction affects the Flags accordingly: Z = 1, S = 0, V = 0.*

Register FCh (Flags)
Flag Register (Read/Write)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

User Flag (F1)
User Flag (F2)
Half Carry Flag (H)
Decimal Adjust Flag (D)
Overflow Flag (V)
Sign Flag (S)
Zero Flag (Z)
Carry Flag (C)
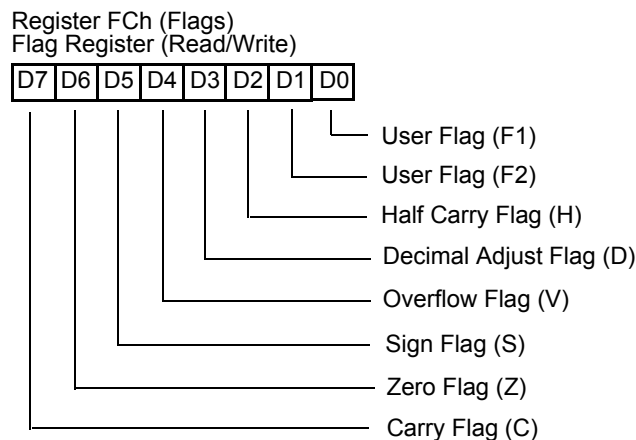
Figure 133. Z8 Flag Register

Carry Flag

The Carry Flag (C) is set to 1 whenever the result of an arithmetic operation generates a carry or a borrow the high order bit 7. Otherwise, the Carry Flag is cleared to 0.

Following Rotate and Shift instructions, the Carry Flag contains the last value shifted out of the specified register.

An instruction can set, reset, or complement the Carry Flag.

IRET may change the value of the Carry Flag when the Flag Register, saved in the Stack, is restored.

Zero Flag

For arithmetic and logical operations, the Zero Flag (Z) is set to 1 if the result is zero. Otherwise, the Zero Flag is cleared to 0.

If the result of testing bits in a register is `00h`, the Zero Flag is set to 1. Otherwise the Zero Flag is cleared to 0.

If the result of a Rotate or Shift operation is `00h`, the Zero Flag is set to 1. Otherwise, the Zero Flag is cleared to 0.

IRET changes the value of the Zero Flag when the Flag Register saved in the Stack is restored. The WDT Instruction sets the Zero Flag to a 1.

Sign Flag

The Sign Flag (S) stores the value of the most significant bit of a result following an arithmetic, logical, Rotate, or Shift operation.

When performing arithmetic operations on signed numbers, binary two's-complement notation is used to represent and process information. A positive number is identified by a 0 in the most significant bit position (bit 7); therefore, the Sign Flag is also 0.

A negative number is identified by a 1 in the most significant bit position (bit 7); therefore, the Sign Flag is also 1.

IRET changes the value of the Sign Flag when the Flag Register saved in the Stack is restored.

Overflow Flag

For signed arithmetic, Rotate, and Shift operations, the Overflow Flag (V) is set to 1 when the result is greater than the maximum possible number (> 127) or less than the minimum possible number (< -128) that can be represented in two's-complement form. The Overflow Flag is set to 0 if no overflow occurs.

Following logical operations the Overflow Flag is set to 0.

IRET changes the value of the Overflow Flag when the Flag Register saved in the Stack is restored.

Decimal Adjust Flag

The Decimal Adjust Flag (D) is used for BCD arithmetic. Because the algorithm for correcting BCD operations is different for addition and subtraction, this flag specifies what type of instruction was last executed so that the subsequent Decimal Adjust (DA) operation can function properly. Normally, the Decimal Adjust Flag cannot be used as a test condition.

After a subtraction, the Decimal Adjust Flag is set to 1. Following an addition it is cleared to 0.

IRET changes the value of the Decimal Adjust Flag when the Flag Register saved in the Stack is restored.

Half Carry Flag

The Half Carry Flag (H) is set to 1 whenever an addition generates a carry bit 3 (Overflow) or a subtraction generates a borrow bit 3. The Half Carry Flag is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. As in the case of the Decimal Adjust Flag, This Flag is normally not accessed.

IRET changes the value of the Half Carry Flag when the Flag Register saved in the Stack is restored.

Condition Codes

The C, Z, S, and V Flags control the operation of the 'Conditional' Jump instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms bits 4-7 of the conditional instructions.

Table 34. Z8 Flag Definitions

Flag	Description
C	Carry Flag
Z	Zero Flag
S	Sign Flag
V	Overflow Flag
D	Decimal Adjust Flag
H	Half Carry Flag

Condition codes and Flag settings are summarized in Table 34 through Table 36 on page 147. Notation for the Flags and how they are affected are as follows:

Table 35. Flag Settings Definitions

Symbol	Definition
0	Cleared to 0
1	Set to 1
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Table 36. Condition Codes

Binary	Hex	Mnemonic	Definition	Flag Settings
0000	0	F	Always False	—
1000	8	(blank)	Always True	—
0111	7	C	Carry	C = 1
1111	F	NC	No Carry	C = 0
0110	6	Z	Zero	Z = 1
1110	E	NZ	Non-Zero	Z = 0
1101	D	PL	Plus	S = 0
0101	5	MI	Minus	S = 1
0100	4	OV	Overflow	V = 1
1100	C	NOV	No Overflow	V = 0
0110	6	EQ	Equal	Z = 1
1110	E	NE	Not Equal	Z = 0
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
0001	1	LT	Less Than	(S XOR V) = 1
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1
1111	F	UGE	Unsigned Greater Than or Equal	C = 0
0111	7	ULT	Unsigned Less Than	C = 1
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1

Notation and Binary Encoding

In the detailed instruction descriptions that make up the rest of this chapter, operands and status flags are represented by a notational shorthand. Operands, condition codes, address modes, and their notations are as follows (see Table 37 on page 148).

Table 37. Notational Shorthand

Notation	Address Mode	Operand	Range *
cc	Condition Code		See Condition Codes
r	Working Register	Rn	n = 0–15
R	Register	Reg	Reg. represents a number in the range of 00h to FFh
	or Working Register	Rn	n = 0–15
RR	Register Pair	Reg	Reg. represents an even number in the range of 00h to FEh
	or Working Register Pair	RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
Ir	Indirect Working Register	@Rn	n = 0 –15
IR	Indirect Register	@Reg	Reg. represents a number in the range of 00h to FFh
	or Indirect Working Register	@Rn	n = 0– 15
Irr	Indirect Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
IRR	Indirect Register Pair	@Reg	Reg. represents an even number in the range 00h to FFh
	or Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
X	Indexed	Reg (Rn)	Reg. represents a number in the range of 00h to FFh and n = 0–15
DA	Direct Address	Addrs	Addrs. represents a number in the range of 00h to FFh
RA	Relative Address	Addrs	Addrs. represents a number in the range of +127 to –128 which is an offset relative to the address of the next instruction
IM	Immediate	#Data	Data is a number between 00h to FFh

*See the device product specification to determine the exact register file range available. The register file size varies by the device type.

Additional symbols used are listed in Table 38.

Table 38. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag Register (FCh)
RP	Register Pointer (FDh)
IMR	Interrupt Mask Register (FBh)
#	Immediate Operand Prefix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix
B	Binary Number Suffix
OPC	Opcode

Assignment of a value is indicated by the symbol "←". For example,

```
dst ← dst + src
```

indicates the source data is added to the destination data and the result is stored in the destination location.

The notation 'addr(n)' is used to refer to bit'n' of a given location. For example,

```
dst (7)
```

refers to bit 7 of the destination operand.

Assembly Language Syntax

For proper instruction execution, Z8 assembly language syntax requires 'dst, src' be specified, in that order. The following instruction descriptions show the format of the object code produced by the assembler. This binary format should be followed if manual program coding is preferred or if you intend to implement your own assembler.

Example: If the contents of registers 43h and 08h are added and the result is stored in 43h, the assembly syntax and resulting object code is:

ASM:	ADD	43h,	08h	(ADD dst, src)
OBJ:	04	08	43	(OPC src, dst)

In general, whenever an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or a Working Register R0–R15. If, in the above example, register `08h` is a Working Register, the assembly syntax and resulting object code would be:

ASM:	ADD	43h,	08h	(ADD dst, src)
OBJ:	04	08	43	(OPC src, dst)

▶ **Note:** *Refer to the device product specification to determine the exact register file range available. The register file size varies by device type.*

Z8® Instruction Summary

Table 39 provides the summary of Z8 instruction set.

Table 39. Summary of Z8 Instruction Set

Instruction and Operation	Address Mode dst	Address Mode src	Op Code Byte (Hex)	C	Z	S	V	D	H
ADC dst, src dst ← dst + src +C	†		1[ ]	[	[	[	[	0	[
ADD dst, src dst ← dst + src	†		0[ ]	[	[	[	[	0	[
AND dst, src dst ← dst AND src	†		5[ ]	–	[	[	0	–	–
CALL dst SP ← SP–2 and PC ← dst or @ SP ← PC	DA IRR		D6 D4	–	–	–	–	–	–
CCF C ←NOT C			EF	[	–	–	–	–	–
CLR dst dst ← 0	R IR		B0 B1	–	–	–	–	–	–
COM dst dst ← NOT dst	R IR		60 61	–	[	[	0	–	–

Table 39. Summary of Z8 Instruction Set (Continued)

Instruction and Operation	Address Mode dst	src	Op Code Byte (Hex)	C	Z	S	V	D	H
CP dst, src dst − src	†		A[ ]	[	[	[	[	–	–
DA dst	R		40	[	[	[	X	–	–
dst ← DA dst	IR		41						
DEC dst	R		00	–	[	[	[	–	–
dst ← dst–1	IR		01						
DECW dst	R R		80	–	[	[	[	–	–
dst ← dst–1	IR		81						
DI dst IMR(7) ← 0			8 F	–	–	–	–	–	–
DJNZr, dst r ← r–1 if r ≠ 0 PC ← PC + dst Range:+127, –128	RA		rA r = 0-F	–	–	–	–	–	–
EI IMR(7) ← 1			9 F	–	–	–	–	–	–
HALT			7 F	–	–	–	–	–	–
INC dst dst ← dst + 1	r		rE r = 0-F	–	[	[	[	–	–
	R		20						
	IR		21						
INCW dst	RR		A0	–	[	[	[	–	–
dst ← dst + 1	IR		A1						

Instruction and Operation	Address Mode dst	src	Op Code Byte (Hex)	C	Z	S	V	D	H
IRET FLAGS←@SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; and IMR(7)–1			B F	[	[	[	[	[	[

Table 39. Summary of Z8 Instruction Set (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Op Code Byte (Hex)	Flags Affected C	Z	S	V	D	H
JP cc, dst	DA		cD	–	–	–	–	–	–
if cc is true,			c = 0–F						
then PC ← dst	IRR		30						
JR cc, dst	RA		cB	–	–	–	–	–	–
if cc is true, PC ← PC + dst Range: +127, –128			c = 0–F						
LD dst, src	r	Im	r C	–	–	–	–	–	–
dst ← src	r	R	r 8						
	R	r	r 9						
			r = 0–F						
	r	X	C 7						
	X	r	D 7						
	r	Ir	E 3						
	Ir	r	F 3						
	R	R	E 4						
	R	IR	E 5						
	R	IM	E 6						
	IR	IM	E 7						
	IR	R	F 5						
LDC dst, src	r	Irr	C 2	–	–	–	–	–	–
dst ← src	Irr	r	D 2						
LDCI dst, src	Ir	Irr	C 3	–	–	–	–	–	–
dst ← src r ← r + 1 or rr ←rr + 1	Irr	Ir	D 3						
LDE dst, src	r	Irr	82	–	–	–	–	–	–
dst ← src	Irr	r	92						
LDEI dst, src	Ir	Irr	83	–	–	–	–	–	–

Table 39. Summary of Z8 Instruction Set (Continued)
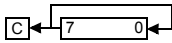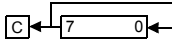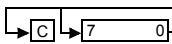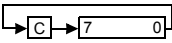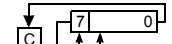
Instruction and Operation	Address Mode dst	src	Op Code Byte (Hex)	Flags Affected C	Z	S	V	D	H
dst ← src and r ← r + 1 or rr ←rr + 1	Irr	Ir	93						
NOP			FF	–	–	–	–	–	–
OR dst, src dst ← dst OR src	†		4[ ]	–	[	[	0	–	–
POP dst	R		50	–	–	–	–	–	–
dst ← @SP and SP ← SP + 1	IR		51						
PUSH src	R		70	–	–	–	–	–	–
SP ← SP–1 and @SP ← src	IR		71						
RCF C ← 0			C F	0	–	–	–	–	–
RET PC ← @SP; SP ← SP + 2			A F	–	–	–	–	–	–
RL dst	R IR		90 91	[	[	[	[	–	–
RLC dst	R IR		10 11	[	[	[	[	–	–
RR dst	R IR		E 0 E 1	[	[	[	[	–	–
RRC dst	R IR		C 0 C 1	[	[	[	[	–	–
SBC dst, src dst ← dst–src–C	†		3[ ]	[	[	[	[	1	[
SCF C ← 1			D F	1	–	–	–	–	–
SRA dst	R IR		D 0 D 1	[	[	[	0	–	–

Table 39. Summary of Z8 Instruction Set (Continued)

Instruction and Operation	Address Mode dst	src	Op Code Byte (Hex)	Flags Affected C	Z	S	V	D	H
SRP dst RP ← src	Im		31	–	–	–	–	–	–
STOP			6 F	–	–	–	–	–	–
SUB dst, src dst ← dst–src	†		2[ ]	[	[	[	[	1	[
SWAP dst	R IR		F0 F1	X	[	[	X	–	–
TCM dst, src (NOT dst) AND src	†		6[ ]	–	[	[	0	–	–
TM dst, src dst AND src	†		7[ ]	–	[	[	0	–	–
WDh			4 F	–	X	X	X	–	–
WDT			5 F	–	X	X	X	–	–
XOR dst, src dst AND src XOR src	†		7[ ]	–	[	[	0	–	–

SWAP dst diagram:

*Note: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair. For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Table 40 provides a summary of Z8 address modes.

Table 40. Summary of Z8® Address Modes

Address Mode		Lower
dst	**src**	**Op Code Nibble**
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

Op Code Map

Figure 134 displays the opcode map.

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 DEC R1	6.5 DEC IR1	6.5 ADD r1, r2	6.5 ADD r1, Ir2	10.5 ADD R2, R1	10.5 ADD IR2, R1	10.5 ADD R1, IM	10.5 ADD IR1, IM	6.5 LD r1, R2	6.5 LD r2, R1	12/10.5 DJNZ r1, RA	12/10.0 JR cc, RA	6.5 LD r1, IM	12.10.0 JP cc, DA	6.5 INC r1	
1	6.5 RLC R1	6.5 RLC IR1	6.5 ADC r1, r2	6.5 ADC r1, Ir2	10.5 ADC R2, R1	10.5 ADC IR2, R1	10.5 ADC R1, IM	10.5 ADC IR1, IM								
2	6.5 INC R1	6.5 INC IR1	6.5 SUB r1, r2	6.5 SUB r1, Ir2	10.5 SUB R2, R1	10.5 SUB IR2, R1	10.5 SUB R1, IM	10.5 SUB IR1, IM								
3	8.0 JP IRR1	6.1 SRP IM	6.5 SBC r1, r2	6.5 SBC r1, Ir2	10.5 SBC R2, R1	10.5 SBC IR2, R1	10.5 SBC R1, IM	10.5 SBC IR1, IM								
4	8.5 DA R1	8.5 DA IR1	6.5 OR r1, r2	6.5 OR r1, Ir2	10.5 OR R2, R1	10.5 OR IR2, R1	10.5 OR R1, IM	10.5 OR IR1, IM								6.0 WDh
5	10.5 POP R1	10.5 POP IR1	6.5 AND r1, r2	6.5 AND r1, Ir2	10.5 AND R2, R1	10.5 AND IR2, R1	10.5 AND R1, IM	10.5 AND IR1, IM								6.0 WDT
6	6.5 COM R1	6.5 COM IR1	6.5 TCM r1, r2	6.5 TCM r1, Ir2	10.5 TCM R2, R1	10.5 TCM IR2, R1	10.5 TCM R1, IM	10.5 TCM IR1, IM								6.0 STOP
7	10/12.1 PUSH R2	12/14.1 PUSH IR2	6.5 TM r1, r2	6.5 TM r1, Ir2	10.5 TM R2, R1	10.5 TM IR2, R1	10.5 TM R1, IM	10.5 TM IR1, IM								7.0 HALT
8	10.5 DECW RR1	10.5 DECW IR1	12.0 LDE r1, Irr2	18.0 LDEI Ir1, Irr2												6.1 DI
9	6.5 RL R1	6.5 RL IR1	12.0 LDE r2, Irr1	18.0 LDEI Ir2, Irr1												6.1 EI
A	10.5 INCW RR1	10.5 INCW IR1	6.5 CP r1, r2	6.5 CP r1, Ir2	10.5 CP R2, R1	10.5 CP IR2, R1	10.5 CP R1, IM	10.5 CP IR1, IM								14.0 RET
B	6.5 CLR R1	6.5 CLR IR1	6.5 XOR r1, r2	6.5 XOR r1, Ir2	10.5 XOR R2, R1	10.5 XOR IR2, R1	10.5 XOR R1, IM	10.5 XOR IR1, IM								16.0 IRET
C	6.5 RRC R1	6.5 RRC IR1	12.0 LDC r1, Irr2	18.0 LDCI Ir1, Irr2				10.5 LD r1,x,R2								6.5 RCF
D	6.5 SRA R1	6.5 SRA IR1	12.0 LDC Irr1, r2	18.0 LDCI Irr1, Ir2	20.0 CALL* IRR1		20.0 CALL DA	10.5 LD r2,x,R1								6.5 SCF
E	6.5 RR R1	6.5 RR IR1		6.5 LD r1, IR2	10.5 LD R2, R1	10.5 LD IR2, R1	10.5 LD R1, IM	10.5 LD IR1, IM								6.5 CCF
F	8.5 SWAP R1	8.5 SWAP IR1		6.5 LD Ir1, r2		10.5 LD R2, IR1										6.0 NOP

Bytes per Instruction: 2 | 3 | 2 | 3 | 1
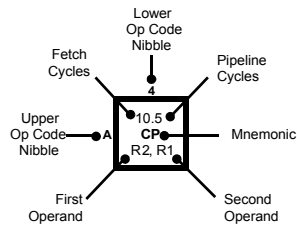
Legend:
R = 8-Bit Address
r = 4-Bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand,
Second Operand

Note: Blank areas are reserved.

*2-byte instruction appears as
a 3-byte instruction

Key box:
Fetch Cycles — Lower Op Code Nibble — Pipeline Cycles
Upper Op Code Nibble — A — 10.5 CP — Mnemonic
R2, R1
First Operand — Second Operand

Figure 134. Op Code Map

Instruction Description

Table 41 provides quick reference to each of the ZTP process manipulation functions.

Table 41. Process Manipulation Functions

Add	Logical OR
Add With Carry	Logical Exclusive OR
Call Procedure	POP
Complement Carry Flag	PUSH
Clear	Reset Carry Flag
Complement	Return
Compare	Rotate Left
Decimal Adjust	Rotate Left Through Carry
Decrement	Rotate Right
Decrement and Jump if Non-Zero	Rotate Right Through Carry
Decrement Word	Set Carry Flag
Disable Interrupts	Set Register Pointer
Enable Interrupts	Shift Right Arithmetic
Halt	Rotate Right Through Carry
Increment	Set Carry Flag
Increment Word	Set Register Pointer
Interrupt Return	Shift Right Arithmetic
Jump	STOP
Jump Relative	Subtract
Load	Subtract With Carry
Load Constant	Swap Nibbles
Load Constant Autoincrement	Test Complement Under Mask
Load External Data	Test Under Mask
Load External Data Autoincrement	Watchdog Timer
No Operation	Watchdog Timer Enable During Halt Mode
Logical AND	

Add

Syntax

```
ADD dst, src
```

Instruction Format

					Address Mode	
		Cycles	**OPC (Hex)**	**dst**	**src**
OPC / dst src			6	02	r	r
			6	03	r	Ir
OPC / src / dst			10	04	R	R
			10	05	R	IR
OPC / dst / src			10	06	R	IM
			10	07	IR	IM

Operation

```
dst ← dst + src
```

The source operand is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not affected.

Flags	Description
C	Set if there is a carry from the most significant bit of the result; cleared otherwise
Z	Set if the result is zero; cleared otherwise
S	Set if the result is negative; cleared otherwise
V	Set if an arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise
D	Always cleared
H	Set if there is a carry from the most significant bit of the low order four bits of the result; cleared otherwise

▸ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	src		or		E	dst

Example 1

If Working Register R3 contains `16h` and Working Register R11 contains `20h`, the statement:

```
ADD R3, R11
Op Code: 02 3B
```

leaves the value `36h` in Working Register R3. The C, Z, S, V, D, and H Flags are all cleared.

Example 2

If Working Register R16 contains `16h`, Working Register R10 contains `20h`, and Register `20h` contains `11h`, the statement:

```
ADD R16, @R10
Op Code: 03 FA
```

leaves the value `27h` in Working Register R16. The C, Z, S, V, D, and H Flags are all cleared.

Example 3

If Register `34h` contains `2Eh` and Register `12h` contains `1Bh`, the statement:

```
ADD 34h, 12h
Op Code: 04 12 34
```

leaves the value `49h` in Register `34h`. The H Flag is set, and the C, Z, S, V, and D Flags are cleared.

Example 4

If Register `4Bh` contains `82h`, Working Register R3 contains `10h`, and Register `10h` contains `01h`, the statement:

```
ADD 3Eh, @R3
Op Code: 05 E3 4B
```

leaves the value `83h` in Register `4Bh`. The S Flag is set, and the C, Z, V, D, and H Flags are cleared.

Example 5

If Register `6Ch` contains `2Ah`, the statement:

```
ADD 6Ch, #03h
Op Code: 06 6C 03
```

leaves the value `2Dh` in Register `6Ch`. The C, Z, S, V, D, and H Flags are all cleared.

Example 6

If Register `D4h` contains `5Fh` and Register `5Fh` contains `4Ch`, the statement:

```
ADD @D4h, #02h
Op Code: 07 D4 02
```

leaves the value `4Eh` in Register `5Fh`. The C, Z, S, V, D, and H Flags are all cleared.

Add With Carry

Syntax

```
ADC dst, src
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst	Address Mode src
OPC	dst | src	6	02	r	r
		6	03	r	lr
OPC	src | dst	10	04	R	R
		10	05	R	IR
OPC	dst | src	10	06	R	IM
		10	07	IR	IM

Operation

```
dst ← dst + src + C
```

The source operand, along with the setting of the Carry (C) Flag, is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not affected. In multiple precision arithmetic, this instruction permits the carry from the addition of low order operands to be carried into the addition of high order operands.

Flag	Description
C	Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z	Set if the result is zero; cleared otherwise.
S	Set if the result is negative; cleared otherwise.
V	Set if an arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D	Always cleared.
H	Set if there is a carry from the most significant bit of the low order four bits of the result; cleared otherwise.

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src		or	E	dst

Example 1

If Working Register R3 contains `16h`, the C Flag is set to 1, and Working Register R11 contains `20h`, the statement:

```
ADC R3, R11
Op Code: 12 3B
```

leaves the value `37h` in Working Register R3. The C, Z, S, V, D, and H Flags are all cleared.

Example 2

If Working Register R16 contains `16h`, the C Flag is not set, Working Register R10 contains `20h`, and Register `20h` contains `11h`, the statement:

```
ADC R16, @R10
Op Code: 13 FA
```

leaves the value `27h` in Working Register R16. The C, Z, S, V, D, and H Flags are all cleared.

Example 3

If Register `34h` contains `2Eh`, the C Flag is set, and Register `12h` contains `1Bh`, the statement:

```
ADC 34h, 12h
Op Code: 14 12 34
```

leaves the value `4Ah` in Register `34h`. The H Flag is set, and the C, Z, S, V, and D Flags are cleared.

Example 4

If Register `4Bh` contains `82h`, the C Flag is set, Working Register R3 contains `10h`, and Register `10h` contains `01h`, the statement:

```
ADC 4Bh, @R3
Op Code: 15 E3 4B
```

leaves the value `84h` in Register `4Bh`. The S Flag is set, and the C, Z, V, D, and H Flags are cleared.

Example 5

If Register `6Ch` contains `2Ah`, and the C Flag is not set, the statement:

```
ADC 6Ch, #03h
Op Code: 16 6C 03
```

leaves the value `2Dh` in Register `6Ch`. The C, Z, S, V, D, and H Flags are all cleared.

Example 6

If Register `D4h` contains `5Fh`, Register `5Fh` contains `4Ch`, and the C Flag is set, the statement:

```
ADC @D4h, #02h
Op Code: 17 D4 02
```

leaves the value `4Fh` in Register `5Fh`. The C, Z, S, V, D, and H Flags are all cleared.

Call Procedure

Syntax

```
CALL dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	20	D6	DA
OPC	dst	20	D4	IRR

Operation

```
SP ← SP-2
@SP ← PC
PC ← dst
```

The Stack pointer is decremented by two, the current contents of the Program Counter (PC) (address of the first instruction following the CALL instruction) are pushed onto the top of the Stack, and the specified destination address is then loaded into the PC. The PC now points to the first instruction of the procedure.

At the end of the procedure a RET (return) instruction can be used to return to the original program flow. RET pops the top of the Stack and replace the original value into the PC.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

▶ **Note:** *Address mode IRR can be used to specify a 4-bit Working Register Pair. In this format, the destination Working Register Pair operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register Pair RR12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src	or	E	dst
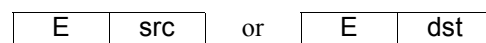
Example 1

If the contents of the PC are 1A47h and the contents of the SP (Registers `FEh` and `FFh`) are 3002h, the statement:

```
CALL 3521h
Op Code: D6 35 21
```

causes the SP to be decremented to `3000h`, `1A4Ah` (the address following the CALL instruction) to be stored in external data memory 3000 and 3001h, and the PC to be loaded with `3521h`. The PC now points to the address of the first statement in the procedure to be executed.

Example 2

If the contents of the PC are 1A47h, the contents of the SP (Register `FFh`) are 72h, the contents of Register A4h are `34h`, and the contents of Register Pair `34h` are `3521h`, the statement:

```
CALL @A4h
Op Code: D4 A4
```

causes the SP to be decremented to `70h`, `1A4Ah` (the address following the CALL instruction) to be stored in R`70h` and `71h`, and the PC to be loaded with `3521h`. The PC now points to the address of the first statement in the procedure to be executed.

Complement Carry Flag

Syntax

```
CCF
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	EF

Operation

```
C ← NOT C
```

The C Flag is complemented. If C = 1, then it is changed to C = 0; or, if C = 0, then it is changed to C = 1.

Flag	Description
C	Complemented
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example

If the C Flag contains a 0, the statement:

```
CCF
Op Code: EF
```

changes the C Flag from C = 0 to C = 1.

Clear

Syntax

```
CLR dst
```

Instruction Format

			Cycles	OPC (Hex)	Address dst
OPC		dst	6	B0	R
			6	B1	IR

Operation

```
dst ← 0
```

The destination operand is cleared to `00h`.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b (`Eh`)` *to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example

If Working Register R6 contains AFh, the statement:

```
CLR R6
Op Code: B0 E6
```

leaves the value `00h` in Working Register R6.

If Register `A5h` contains the value `23h`, and Register `23h` contains the value `FCh`, the statement:

```
CLR @A5h
Op Code: B1 A5
```

leaves the value `00h` in Register `23h`.

Complement

Syntax

```
COM dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	60	R
		6	61	IR

Operation

```
dst ← NOT dst
```

The contents of the destination operand are complemented (one's complement). All 1 bits are changed to 0, and all 0 bits are changed to 1.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise.
S	Set if result bit 7 is set; cleared otherwise.
V	Always reset to 0.
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If Register `08h` contains `24h` (`00100100b`), the statement:

```
COM 08h
Op Code: 60 08
```

leaves the value `DBh` (`11011011`) in Register `08h`. The S Flag is set, and the Z and V Flags are cleared.

Example 2

If Register `08h` contains `24h`, and Register `24h` contains `FFh` (`11111111b`), the statement:

```
COM @08h
Op Code: 61 08
```

leaves the value `00h` (`00000000b`) in Register `24h`. The Z Flag is set, and the V and S Flags are cleared.

Compare

Syntax

```
CP dst, src
```

Instruction Format

				Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src		6	A2	r	r
				6	A3	r	Ir
OPC	src	dst		10	A4	R	R
				10	A5	R	IR
OPC	dst	src		10	A6	R	IM
				10	A7	IR	IM

Operation

```
dst–src
```

The source operand is compared to (subtracted from) the destination operand, and the appropriate Flags are set accordingly. The contents of both operands are unaffected.

Flag	Description
C	Cleared if there is a carry from the most significant bit of the result. Set otherwise indicating a borrow.
Z	Set if the result is zero; cleared otherwise.
S	Set if result bit 7 is set (negative); cleared otherwise.
V	Set if arithmetic overflow occurs; cleared otherwise.
D	Unaffected.
H	Unaffected.

Note: *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

Example 1

E	src	or	E	dst
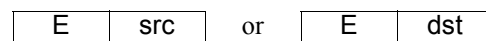
If Working Register R3 contains `16h` and Working Register R11 contains `20h`, the statement:

```
CP R3, R11
Op Code: A2 3B
```

sets the C and S Flags, and the Z and V Flags are cleared.

Example 2

If Working Register R15 contains `16h`, Working Register R10 contains `20h`, and Register `20h` contains `11h`, the statement:

```
CP R16, @R10
Op Code: A3 FA
```

clears the C, Z, S, and V Flags.

Example 3

If Register `34h` contains `2Eh` and Register `12h` contains `1Bh`, the statement:

```
CP 34h,12h
Op Code: A4 12 34
```

clears the C, Z, S, and V Flags.

Example 4

If Register `4Bh` contains `82h`, Working Register R3 contains `10h`, and Register `10h` contains `01h`, the statement:

```
CP 4Bh, @R3
Op Code: A5 E3 4B
```

sets the S Flag, and clears the C, Z, and V Flags.

Example 5

If Register `6Ch` contains `2Ah`, the statement:

```
CP 6Ch, #2Ah
Op Code: A6 6C 2A
```

sets the Z Flag, and the C, S, and V Flags are all cleared.

Example 6

If Register `D4h` contains `FCh`, and Register `FCh` contains `8Fh`, the statement:

```
CP @D4h, 7Fh
Op Code: A7 D4 FF
```

sets the V Flag, and the C, Z, and S Flags are all cleared.

Decimal Adjust

Syntax

```
DA dst
```

Instruction Format

			OPC	Address Mode
		Cycles	(Hex)	dst
OPC	dst	8	40	R
		8	41	IR

Operation

```
dst ← DA dst
```

The destination operand is adjusted to form two 4-bit BCD digits following a binary addition or subtraction operation on BCD encoded bytes. For addition (ADD and ADC) or subtraction (SUB and SBC), the following table indicates the operation performed.

Instruction	Carry Before DA	Bits 7–4 Value (Hex)	H Flag Before DA	Bits 3–0 Value (Hex)	Number Added To Byte	Carry After DA
	0	0–9	0	0–9	00	0
	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
ADD	0	A–F	0	0–9	60	1
ADC	0	9–F	0	A–F	66	1
	0	A–F	1	0–3	66	1
	1	0–2	0	0–9	60	1
	1	0–2	0	A–F	66	1
	1	0–3	1	0–3	66	1
	0	0–9	0	0–9	00	0
SUB	0	0–8	1	6–F	FA	0
SBC	1	7–F	0	0–9	A0	1
	1	6–F	1	6–F	9A	1

If the destination operand is not the result of a valid addition or subtraction of BCD digits, the operation is undefined.

Flag   Description

C   Set if there is a carry from the most significant bit; cleared
        otherwise (see table above).

Z   Set if the result is zero; cleared otherwise.

S   Set if result bit 7 is set (negative); cleared otherwise.

D   Unaffected

H   Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If addition is performed using the BCD values 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic.

```
  0001 0101 = 15h
+ 0010 0111 = 27h
  0011 1100 = 3Ch
```

If the result of the addition is stored in Register `5Fh`, the statement:

```
DA 5Fh
Op Code: 40 5F
```

adjusts this result so the correct BCD representation is obtained.

```
0011 1100 = 3Ch
0000 0110 = 06h
0100 0010 = 42h
```

Register `5Fh` now contains the value `42h`. The C, Z, and S Flags are cleared, and V is undefined.

Example 2

If addition is performed using the BCD values 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic.

```
  0001 0101 = 15h
+ 0010 0111 = 27h
```

```
    0011 1100 = 3Ch
```

Register `45F` contains the value `5Fh`, and the result of the addition is stored in Register `5Fh`, the statement:

```
DA @45h
Op Code: 40 45
```

adjusts this result so the correct BCD representation is obtained.

```
0011 1100 = 3Ch
0000 0110 = 06h
0100 0010 = 42h
```

Register `5Fh` now contains the value `42h`. The C, Z, and S Flags are cleared, and V is undefined.

Decrement

Syntax

```
DEC dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	00	R
		6	01	IR

Operation

```
dst ← dst-1
```

The contents of the destination operand are decremented by one.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise
S	Set if the result of bit 7 is set (negative); cleared otherwise
V	Set if arithmetic overflow occurs; cleared otherwise
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If Working Register R10 contains 2A%, the statement:

```
DEC R10
Op Code: 00 EA
```

leaves the value `29h` in Working Register R10. The Z, V, and S Flags are cleared.

Example 2

If Register `B3h` contains `CBh`, and Register `CBh` contains `01h`, the statement:

```
DEC @B3h
Op Code: 01 B3
```

leaves the value `00h` in Register `CBh`. The Z Flag is set, and the V and S Flags are cleared.

Decrement and Jump if Non-Zero

Syntax

```
DJNZ r, dst
```

Instruction Format

			Cycles	**OPC (Hex)**	**Address Mode** **dst**
r	OPC	dst	12 If jump taken	rA	r
			10 if jump not taken	(R = 0 to F)	r

Operation

```
r ← r-1;
If r <> 0, PC ← PC + dst
```

The specified Working Register being used as a counter is decremented. If the contents of the specified Working Register are not zero after decrementing, then the relative address is added to the Program Counter (PC) and control passes to the statement whose address is now in the PC. The range of the relative address is +127 to –128. The original value of the PC is the address of the instruction byte following the DJNZ statement. When the specified Working Register counter reaches zero, control falls through to the statement following the DJNZ instruction.

Flag	**Description**
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *The Working Register being used as a counter must be one of the Registers from* 04h *to* EFh. *Use of one of the I/O ports, control or peripheral registers will have undefined results.*

Example

DJNZ is typically used to control a *loop* of instructions. In this example, 12 bytes are moved from one buffer area in the register file to another, follow the steps below:

- Load 12 into the counter (Working Register R6)

- Set up the loop to perform the moves

- End the loop with DJNZ
  The assembly listing required for this routine is as follows:

```
      LD R6, 12      ;Load Counter
LOOP: LD R9, @R6     ;Move one byte to
      LD @R6, R9     ;new location
      DJNZ R6, LOOP  ;Decrement and Loop until counter
                     ;= 0
```

Decrement Word

Syntax

```
DECW dst
```

Instruction Format

		OPC	Address Mode
	Cycles	(Hex)	dst

OPC	dst

| 10 | 80 | RR |
| 10 | 81 | IR |

Operation

```
dst ← dst-1
```

The contents of the destination (which must be an even address) operand are decremented by one. The destination operand can be a Register Pair or a Working Register Pair.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise
S	Set if the result of bit 7 is set (negative); cleared otherwise
V	Set if arithmetic overflow occurs; cleared otherwise
D	Unaffected
H	Unaffected

> **Note:** *Address modes RR or IR can be used to specify a 4-bit Working Register Pair. In this format, the destination Working Register Pair operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register Pair R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If Register Pair `30h` and `31h` contain the value `0AF2h`, the statement:

```
DECW 30h
Op Code: 80 30
```

leaves the value `0AF1h` in Register Pair `30h` and `31h`. The Z, V, and S Flags are cleared.

Example 2

If Working Register R0 contains `30h` and Register Pairs `30h` and `31h` contain the value `FAF3h`, the statement:

```
DECW @R0
Op Code: 81 E0
```

leaves the value `FAF2h` in Register Pair `30h` and `31h`. The S Flag is set, and the Z and V Flags are cleared.

Disable Interrupts

Syntax

```
Dl
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	8F

Operation

```
IMR (7) ← 0
```

Bit 7 of Control Register FBh (the Interrupt Mask Register) is reset to 0. All interrupts are disabled, although they remain *potentially* enabled. (For example, the Global Interrupt Enable is cleared, but not the individual interrupt level enables.)

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example

If Control Register FBh contains 8Ah (10001010) (interrupts IRQ1 and IRQ3 are enabled), the statement:

```
DI
Op Code: 8F
```

sets Control Register FBh to 0Ah (00001010b) and disables these interrupts.

Enable Interrupts

Syntax

```
EI
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	9F

Operation

```
IMR (7) ← 0
```

Bit 7 of Control Register FBh (the Interrupt Mask Register) is set to 1. This allows potentially enabled interrupts to become enabled.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example

If Control Register FBh contains 0Ah (00001010) (interrupts IRQ1 and IRQ3 are selected), the statement:

```
EI
Op Code: 9F
```

sets Control Register FBh to 8Ah (10001010b) and enables IRQ1 and IRQ3.

Halt

Syntax

```
HALT
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	7F

Operation

The HALT instruction turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and the external interrupts IRQ1, IRQ2, and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *In order to enter HALT mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. You must execute a NOP immediately before the execution of the HALT instruction.*

Example

Assuming the Z8® CPU is in normal operation, the statements:

```
NOP
HALT
Op Codes: FF 7F
```

place the Z8 CPU into HALT mode.

Increment

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
dst	OPC	6	rE	r
OPC	dst	6	20	R
		6	21	IR

Operation

```
dst ← dst + 1
```

The contents of the destination operand are incremented by one.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise.
S	Set if the result of bit 7 is set (negative); cleared otherwise.
V	Set if arithmetic overflow occurs; cleared otherwise.
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If Working Register R10 contains 2Ah, the statement:

```
INC R10
Op Code: AE
```

leaves the value 2Bh in Working Register R10. The Z, V, and S Flags are cleared.

Example 2

If Register B3h contains CBh, the statement:

```
INC B3h
Op Code: 20 B3
```

leaves the value CCh in Register CBh. The S Flag is set, and the Z and V Flags are cleared.

Example 3

If Register `B3h` contains `CBh` and Register `BCh` contains `FFh`, the statement:

```
INC @B3h
Op Code: 21 B3
```

leaves the value `00h` in Register `CBh`. The Z Flag is set, and the V and S Flags are cleared.

Increment Word

Syntax

```
INCW dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	10	A0	RR
		10	A1	IR
		10	A0	R

Operation

```
dst ← dst-1
```

The contents of the destination (which must be an even address) operand is decremented by one. The destination operand can be a Register Pair or a Working Register Pair.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise.
S	Set if the result of bit 7 is set (negative); cleared otherwise.
V	Set if arithmetic overflow occurs; cleared otherwise.
D	Unaffected
H	Unaffected

▶ **Note:** *Address modes RR or IR can be used to specify a 4-bit Working Register Pair. In this format, the destination Working Register Pair operand is specified by adding* 1110b *(Eh) to the high nibble of the operand. For example, if Working Register Pair R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If Register Pairs `30h` and `31h` contain the value `0AF2h`, the statement:

```
INCW 30h
Op Code: A0 30
```

leaves the value `0AF3h` in Register Pair `30h` and `31h`. The Z, V, and S Flags are cleared.

Example 2

If Working Register R0 contains `30h`, and Register Pairs `30h` and `31h` contain the value `FAF3h`, the statement:

```
INCW @R0
Op Code: A1 E0
```

leaves the value `FAF4h` in Register Pair `30h` and `31h`. The S Flag is set, and the Z and V Flags are cleared.

Interrupt Return

Syntax

```
IRET
```

Instruction Format

	Cycles	OPC (Hex)
OPC	16	8F

Operation

```
FLAGS ← @SP
SP ← SP + 1
PC ← @SP
SP ← SP + 2
IMR (7) ← 1
```

This instruction is issued at the end of an interrupt service routine. It restores the Flag Register (Control Register FCh) and the PC. It also re-enables any interrupts that are potentially enabled.

Flag	Description
C	Restored to original setting before the interrupt occurred.
Z	Restored to original setting before the interrupt occurred.
S	Restored to original setting before the interrupt occurred.
V	Restored to original setting before the interrupt occurred.
D	Restored to original setting before the interrupt occurred.
H	Restored to original setting before the interrupt occurred.

Example

If Stack Pointer Low Register FFh currently contains the value 45h, Register 45h contains the value 00h, Register 46h contains 6Fh, and Register 47 contains E4h, the statement:

```
IRET
Op Code: BF
```

restores the FLAG Register FCh with the value 00h, restores the PC with the value 6FE4h, re-enables the interrupts, and sets the Stack Pointer Low to 48h. The next instruction to be executed is at location 6FE4h.

Jump

Syntax

```
JP cc, dst
```

Instruction Format

			Cycles		OPC (Hex)	Address Mode dst
cc	OPC	dst	12	If jump taken	ccD	DA
			10	If jump not taken	cc = 0 to F	
dst		dst	8		30	IRR

Operation

```
If cc (condition code) is true, then PC ← dst
```

A conditional jump transfers Program Control to the destination address if the condition specified by cc (condition code) is true. Otherwise, the instruction following the JP instruction is executed. See Condition Codes on page 146 for a list of condition codes.

The unconditional jump simply replaces the contents of the Program Counter with the contents of the register pair specified by the destination operand. Program Control then passes to the instruction addressed by the PC.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *Address mode IRR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the Carry Flag is set, the statement:

```
JP C, 1520h
Op Code: 7D 15 20
```

replaces the contents of the Program Counter with 1520h and transfers program control to that location. If the Carry Flag had not been set, control would have fallen through to the statement following the JP instruction.

Example 2

If Working Register Pair RR2 contains the value `3F45h`, the statement:

```
JP @RR2
Op Code: 30 E2
```

replaces the contents of the PC with the value `3F45h` and transfers program control to that location.

Jump Relative

Syntax

```
JR cc, dst
```

Instruction Format

	Cycles	OPC (Hex)	Address Mode dst
cc OPC dst	12 If jump taken	ccB	RR
	10 if jump not taken	cc = 0 to F	

Operation

```
If cc is true, PC ← PC + dst
```

If the condition specified by the *cc* is true, the relative address is added to the PC and control passes to the instruction located at the address specified by the PC (see Condition Codes on page 146 for a list of condition codes). Otherwise, the instruction following the JR instruction is executed. The range of the relative address is +127 to –128, and the original value of the PC is taken to be the address of the first instruction byte following the JR instruction.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example 1

If the result of the last arithmetic operation executed is negative, the next four statements (which occupy a total of seven bytes) are skipped with the statement:

```
JR Ml, #9
Op Code: 5B 09
```

If the result was not negative, execution would have continued with the instruction following the JR instruction.

Example 2

A short form of a jump –45 is:

```
JR #−45
Op Code: 8B D3
```

The condition code is *blank* in this case, and is assumed to be *always true*.

Load

Syntax

```
LD dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
				OPC	**Address Mode**	
			Cycles	**(Hex)**	**dst**	**src**
dst OPC	src		6	rC	r	IM
			6	r8	r	R
src OPC	dst		6	r9	R*	r
				r = 0 to F		
OPC	dst OPC		6	E3	r	Ir
			6	F3	Ir	r
OPC	src	dst	10	E4	R	R
			10	E5	R	IR
OPC	dst	src	10	E6	R	IM
			10	E7	IR	IM
OPC	src	dst	10	F5	IR	R
OPC	dst X	src	10	C7	r	X
OPC	src X	dst	10	D7	X	r

Operation

```
dst ← src
```

The contents of the source operand are loaded into the destination operand. The contents of the source operand are not affected.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> ❯ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src		or		E	dst
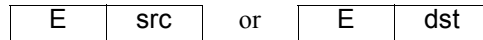
Example 1

The statement:

```
LD R15, #34h
Op Code: FC 34
```

loads the value 34h into Working Register R15.

Example 2

If Register 34h contains the value FCh, the statement:

```
LD R14, 34h
Op Code: F8 34
```

loads the value FCh into Working Register R15. The contents of Register 34h are not affected.

Example 3

If Working Register R14 contains the value 45h, the statement:

```
LD 34h, R14
Op Code: E9 34
```

loads the value 45h into Register 34h. The contents of Working Register R14 are not affected.

Example 4

If Working Register R12 contains the value 34h, and Register 34h contains the value FFh, the statement:

```
LD R13, @R12
Op Code: E3 DC
```

loads the value FFh into Working Register R13. The contents of Working Register R12 and Register R34 are not affected.

Example 5

If Working Register R13 contains the value 45h, and Working Register R12 contains the value 00h the statement:

```
LD @R13, R12
Op Code: F3 DC
```

loads the value `00h` into Register `45h`. The contents of Working Register R12 and Working Register R13 are not affected.

Example 6

If Register `45h` contains the value `CFh`, the statement:

```
LD 34h, 45h
Op Code: E4 45 34
```

loads the value `CFh` into Register `34h`. The contents of Register `45h` are not affected.

Example 7

If Register `45h` contains the value `CFh` and Register `CFh` contains the value `FFh`, the statement:

```
LD 34h, @45h
Op Code: E5 45 34
```

loads the value `FFh` into Register `34h`. The contents of Register `45h` and Register `CFh` are not affected.

Example 8

The statement:

```
LD 34h, #A4h
Op Code: E6 34 A4
```

loads the value A4h into Register `34h`.

Example 9

If Working Register R14 contains the value `7Fh`, the statement:

```
LD @R14, #FCh
Op Code: E7 EE FC
```

loads the value `FCh` into Register `7Fh`. The contents of Working Register R14 are not affected.

Example 10

If Register `34h` contains the value `CFh` and Register `45h` contains the value `FFh`, the statement:

```
LD @34h, 45h
Op Code: F5 45 34
```

loads the value `FFh` into Register `CFh`. The contents of Register `34h` and Register `45h` are not affected.

Example 11

If Working Register R0 contains the value `08h` and Register `2Ch` (`24h` + `08h` = `2Ch`) contains the value `4Fh`, the statement:

```
LD R10, 24h(R0)
Op Code: C7 A0 24
```

loads Working Register R10 with the value `4Fh`. The contents of Working Register R0 and Register `2Ch` are not affected.

Example 12

If Working Register R0 contains the value `0Bh` and Working Register R10 contains `83h` the statement:

```
LD F0h(R0), R10
Op Code: D7 A0 F0
```

loads the value `83h` into Register `FBh` (`F0h` + `0Bh` = `FBh`). Because this is the Interrupt Mask Register, the LOAD statement has the effect of enabling IRQ0 and IRQ1. The contents of Working Registers R0 and R10 are unaffected by the load.

Load Constant

Syntax

```
LDC dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	Address Mode src
OPC	dst	src	12	C2	r	Irr
OPC	dst	src	12	D2	Irr	r

Operation

```
dst ← src
```

This instruction is used to load a byte constant from Program Memory into a Working Register, or vice versa. The address of the Program Memory location is specified by a Working Register Pair. The contents of the source operand are not affected.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example 1

If Working Register Pair R6 and R7 contain the value 30A2h and Program Memory location 30A2h contains the value 22h, the statement:

```
LDC R2, @RR6
Op Code: C2 26
```

loads the value 22h into Working Register R2. The value of Program Memory location 30A2h is unchanged by the load.

Example 2

If Working Register R2 contains the value 22h, and Working Register Pair R6 and R7 contains the value 10A2h, the statement:

```
LDC @RR6, R2
Op Code: D2 26
```

loads the value `22h` into Program Memory location `10A2h`. The value of Working Register R2 is unchanged by the load.

▶ **Note:** *This instruction format is valid only for MCUs which can address external program memory.*

Load Constant Autoincrement

Syntax

```
LDCI dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	18	C3	Ir	Irr
OPC	dst	src	18	D3	Irr	Ir

Operation

```
dst ← src
r ← r + 1
rr ← rr + 1
```

This instruction is used for block transfers of data between Program Memory and the Register File. The address of the Program Memory location is specified by a Working Register Pair, and the address of the Register File location is specified by Working Register. The contents of the source location are loaded into the destination location. Both addresses in the Working Registers are then incremented automatically. The contents of the source operand are not affected.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example 1

If Working Register Pair R6–R7 contains 30A2h, Program Memory location 30A2h and 30A3h contain 22h and BCh respectively, and Working Register R2 contains 20h, the statement:

```
LDCI @R2, @RR6
Op Code: C3 26
```

loads the value 22h into Register 20h. Working Register Pair RR6 is incremented to 30A3h and Working Register R2 is incremented to 21h. A second

```
LDCI @R2, @RR6
Op Code: C3 26
```

loads the value BCh into Register 21h. Working Register Pair RR6 is incremented to 30A4h and Working Register R2 is incremented to 22h.

Example 2

If Working Register R2 contains 20h, Register 20h contains 22h, Register 21h contains BCh, and Working Register Pair R6–R7 contains 30A2h, the statement:

```
LDCI @RR6, @R2
Op Code: D3 26
```

loads the value 22h into Program Memory location 30A2h. Working Register R2 is incremented to 21h and Working Register Pair R6–R7 is incremented to 30A3h. A second

```
LDCI @RR6, @R2
Op Code: D3 26
```

loads the value BCh into Program Memory location 30A3h. Working Register R2 is incremented to 22h and Working Register Pair R6–R7 is incremented to 30A4h.

Load External Data

Syntax

```
LDE dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	12	82	r	Irr
OPC	src	dst	12	92	Irr	r

Operation

```
dst ← src
```

This instruction is used to load a byte from external data memory into a Working Register or vice versa. The address of the external data memory location is specified by a Working Register Pair. The contents of the source operand are not affected.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example 1

If Working Register Pair R6 and R7 contain the value `40A2h` and external data memory location `40A2h` contains the value `22h`, the statement:

```
LDE R2, @RR6
Op Code: 82 26
```

loads the value `22h` into Working Register R2. The value of external data memory location `40A2h` is unchanged by the load.

Example 2

If Working Register Pair R6 and R7 contain the value `404Ah` and Working Register R2 contains the value `22h`, the statement:

```
LDE @RR6, R2
Op Code: 92 26
```

loads the value `22h` into external data memory location `404Ah`

▶ **Note:** *This instruction format is valid only for MCUs which can address external data memory.*

Load External Data Autoincrement

Syntax

```
LDEI dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	18	83	Ir	Irr
OPC	src	dst	18	93	Irr	Ir

Operation

```
dst ← src
r ← r + 1
rr ← rr + 1
```

This instruction is used for block transfers of data between external data memory and the Register File. The address of the external data memory location is specified by a Working Register Pair, and the address of the Register File location is specified by a Working Register. The contents of the source location are loaded into the destination location. Both addresses in the Working Registers are then incremented automatically. The contents of the source are not affected.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example 1

If Working Register Pair R6 and R7 contains 404Ah, external data memory location 404Ah and 404Bh contain ABh and C3h respectively, and Working Register R2 contains 22h, the statement:

```
LDEI @R2, @RR6
Op Code: 83 26
```

loads the value ABh into Register 22h. Working Register Pair RR6 is incremented to 404Bh and Working Register R2 is incremented to 23h. A second

```
LDEI @R2, @RR6
Op Code: 83 26
```

loads the value C3h into Register 23h. Working Register Pair RR6 is incremented to 404Ch and Working Register R2 is incremented to 24h.

Example 2

If Working Register R2 contains 22h, Register 22h contains ABh, Register 23h contains C3h, and Working Register Pair R6 and R7 contains 404Ah, the statement:

```
LDEI @RR6, @R2
Op Code: 93 26
```

loads the value ABh into external data memory location 404Ah. Working Register R2 is incremented to 23h and Working Register Pair RR6 is incremented to 404Bh. A second

```
LDEI @RR6, @R2
Op Code: 93 26
```

loads the value C3h into external data memory location 404Bh. Working Register R2 is incremented to 24h and Working Register Pair RR6 is incremented to 404Ch.

> **Note:** *This instruction format is valid only for MCUs which can address external data memory.*

No Operation

Syntax

```
NOP
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	FF

Operation

No action is performed by this instruction. It is typically used for timing delays or clearing the pipeline.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Logical AND

Syntax

```
AND dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	6	42	r	r
			6	43	r	Ir
OPC	src	dst	10	44	R	R
			10	45	R	IR
OPC	dst	src	10	46	R	IM
			10	47	IR	IM

Operation

```
dst ← dst AND src
```

The source operand is logically ANDed with the destination operand. The AND operation results in a 1 being stored whenever the corresponding bits in the two operands are both 1, otherwise a 0 is stored. The result is stored in the destination operand. The contents of the source bit are not affected.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise
S	Set if the result of bit 7 is set; cleared otherwise
V	Always reset to 0
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R1 contains `34h` (`00111000b`) and Working Register R14 contains `4Dh` (`10001101`), the statement:

```
AND R1, R14
Op Code: 52 1E
```

Example 2

If Working Register R4 contains `F9h` (`11111001b`), Working Register R13 contains `7Bh`, and Register `7Bh` contains `6Ah` (`01101010b`), the statement:

```
AND R4, @R13
Op Code: 53 4D
```

leaves the value `68h` (`01101000b`) in Working Register R4. The Z, V, and S Flags are cleared.

Example 3

If Register `3Ah` contains the value `F5h` (`11110101b`) and Register `42h` contains the value `0Ah` (`00001010`), the statement:

```
AND 3Ah, 42h
Op Code: 54 42 3A
```

leaves the value `00h` (`00000000b`) in Register `3Ah`. The Z Flag is set, and the V and S Flags are cleared.

Example 4

If Working Register R5 contains `F0h` (`11110000b`), Register `45h` contains `3Ah`, and Register `3Ah` contains `7Fh` (`01111111b`), the statement:

```
AND R5, @45h
Op Code: 55 45 E5
```

leaves the value `70h` (`01110000b`) in Working Register R5. The Z, V, and S Flags are cleared.

Example 5

If Register `7Ah` contains the value `F7h` (`11110111b`), the statement:

```
AND 7Ah, #F0h
Op Code: 56 7A F0
```

leaves the value `F0h` (`11110000b`) in Register `7Ah`. The S Flag is set, and the Z and V Flags are cleared.

Example 6

If Working Register R3 contains the value `3Eh` and Register `3Eh` contains the value `ECh` (`11101100b`), the statement:

```
AND @R3, #05h
Op Code: 57 E3 05
```

leaves the value `04h` (`00000100b`) in Register `3Eh`. The Z, V, and S Flags are cleared.

Logical OR

Syntax

```
OR dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst | src		6	02	r	r
			6	03	r	Ir
OPC	src	dst	10	04	R	R
			10	05	R	IR
OPC	dst	src	10	06	R	IM
			10	07	IR	IM

Operation

```
dst ← dst OR src
```

The source operand is logically ORed with the destination operand and the result is stored in the destination operand. The contents of the source operand are not affected. The OR operation results in a one bit being stored whenever either of the corresponding bits in the two operands is a one. Otherwise, a zero bit is stored.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise
S	Set if the result of bit 7 is set; cleared otherwise
V	Always reset to 0
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (*`Ch`*) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R1 contains `34h` (`00111000b`) and Working Register R14 contains `4Dh` (`10001101`), the statement:

```
OR R1, R14
Op Code: 42 1E
```

leaves the value `BDh` (`10111101b`) in Working Register R1. The S Flag is set, and the Z and V Flags are cleared.

Example 2

If Working Register R4 contains `F9h` (`11111001b`), Working Register R13 contains `7Bh`, and Register 7B contains `6Ah` (`01101010b`), the statement:

```
OR R4, @R13
Op Code: 43 4D
```

leaves the value `FBh` (`11111011b`) in Working Register R4. The S Flag is set, and the Z and V Flags are cleared.

Example 3

If Register `3Ah` contains the value `F5h` (`11110101b`) and Register `42h` contains the value `0Ah` (`00001010`), the statement:

```
OR 3Ah, 42h
Op Code: 44 42 3A
```

leaves the value `FFh` (`11111111b`) in Register `3Ah`. The S Flag is set, and the Z and V Flags are cleared.

Example 4

If Working Register R5 contains `70h` (`01110000b`), Register `45h` contains `3Ah`, and Register `3Ah` contains `7Fh` (`01111111b`), the statement:

```
OR R5, @45h
Op Code: 45 45 E5
```

leaves the value `7Fh` (`01111111b`) in Working Register R5. The Z, V, and S Flags are cleared.

Example 5

If Register `7Ah` contains the value `F3h` (`11110111b`), the statement:

```
OR 7Ah, #F0h
Op Code: 46 7A F0
```

leaves the value `F3h` (`11110111b`) in Register `7Ah`. The S Flag is set, and the Z and V Flags are cleared.

Example 6

 If Working Register R3 contains the value `3Eh` and Register `3Eh` contains the value `0Ch` (`00001100b`), the statement:

```
OR @R3, #05h
Op Code: 57 E3 05
```

leaves the value `0Dh` (`00001101b`) in Register `3Eh`. The Z, V, and S Flags are cleared.

Logical Exclusive OR

Syntax

```
XOR dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	6	82	r	r
			6	83	r	Ir
OPC	src	dst	10	84	R	R
			10	85	R	IR
OPC	dst	src	10	86	R	IM
			10	87	IR	IM

Operation

```
dst ← dst XOR src
```

The source operand is logically EXCLUSIVE ORed with the destination operand. The XOR operation results in a 1 being stored in the destination operand whenever the corresponding bits in the two operands are different, otherwise a 0 is stored. The contents of the source operand are not affected.

C   Unaffected
Z   Set if the result is zero; cleared otherwise.
S   Set if the result of bit 7 is set; cleared otherwise.
V   Always reset to 0
D   Unaffected
H   Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R1 contains `34h` (`00111000b`) and Working Register R14 contains `4Dh` (`10001101b`), the statement:

```
XOR R1, R14
Op Code: B2 1E
```

leaves the value `BDh` (`10111101b`) in Working Register R1. The Z, and V Flags are cleared, and the S Flag is set.

Example 2

If Working Register R4 contains `F9h` (`11111001b`), Working Register R13 contains `7Bh`, and Register `7Bh` contains `6Ah` (`01101010b`), the statement:

```
XOR R4, @R13
Op Code: B3 4D
```

leaves the value `93h` (`10010011b`) in Working Register R4. The S Flag is set, and the Z, and V Flags are cleared.

Example 3

If Register `3Ah` contains the value `F5h` (`11110101b`) and Register `42h` contains the value `0Ah` (`00001010b`), the statement:

```
XOR 3Ah, 42h
Op Code: B4 42 3A
```

leaves the value `FFh` (`11111111b`) in Register `3Ah`. The S Flag is set, and the C and V Flags are cleared.

Example 4

If Working Register R5 contains `F0h` (`11110000b`), Register `45h` contains `3Ah`, and Register `3Ah` contains `7Fh` (`01111111b`), the statement:

```
XOR R5, @45h
Op Code: B5 45 E5
```

leaves the value `8Fh` (`10001111b`) in Working Register R5. The S Flag is set, and the C and V Flags are cleared.

Example 5

If Register `7Ah` contains the value `F7h` (`11110111b`), the statement:

```
XOR 7Ah, #F0h
Op Code: B6 7A F0
```

leaves the value `07h` (`00000111b`) in Register `7Ah`. The Z, V and S Flags are cleared.

Example 6

If Working Register R3 contains the value `3Eh` and Register `3Eh` contains the value `6Ch` (`01101100b`), the statement:

```
XOR @R3, #05h
Op Code: B7 E3 05
```

leaves the value `69h` (`01101001b`) in Register `3Eh`. The Z, V, and S Flags are cleared.

POP

Syntax

```
POP dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	10	50	R
		10	51	IR

Operation

```
dst ← @SP
SP ← SP + 1
```

The contents of the location specified by the SP (Stack Pointer) are loaded into the destination operand. The SP is then incremented automatically.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the SP (Control Registers `FEh` and `FFh`) contains the value `70h` and Register `70h` contains `44h`, the statement:

```
POP 34h
Op Code: 50 34
```

loads the value `44h` into Register `34h`. After the POP operation, the SP contains `71h`. The contents of Register 70 are not affected.

Example 2

If the SP (Control Registers `FEh` and `FFh`) contains the value `1000h`, external data memory location `1000h` contains `55h`, and Working Register R6 contains `22h`, the statement:

```
POP @R6
Op Code: 51 E6
```

loads the value `55h` into Register `22h`. After the POP operation, the SP contains `1001h`. The contents of Working Register R6 are not affected.

PUSH

Syntax

```
PUSH src
```

Instruction Format

	Cycles	OPC (Hex)	Address Mode dst
	10 Internal Stack	70	
OPC src	12 External Stack		R
	10 Internal Stack		IR
	10 External Stack	71	

Operation

```
SP ← SP–1
@SP ← src
```

The contents of the SP (stack pointer) are decremented by one, then the contents of the source operand are loaded into the location addressed by the decremented SP, thus adding a new element to the stack.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

```
E  dst
```

Example 1

If the SP contains `1001h`, the statement:

```
PUSH FCh
Op Code: 70 FC
```

stores the contents of Register `FCh` (the Flag Register) in location `1000h`. After the PUSH operation, the SP contains `1000h`.

Example 2

If the SP contains `61h` and Working Register R4 contains `FCh`, the statement:

```
PUSH @R4
Op Code: 71 E4
```

stores the contents of Register `FCh` (the Flag Register) in location `60h`. After the PUSH operation, the SP contains `60h`.

Reset Carry Flag

Syntax

```
RCF
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	CF

Operation

```
C ← 0
```

The C Flag is reset to 0, regardless of its previous value.

Flag	Description
C	Reset to 0
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example

If the C Flag is currently set, the statement:

```
RCF
Op Code: CF
```

resets the Carry Flag to 0.

Return

Syntax

```
RET
```

Instruction Format

	Cycles	OPC (Hex)
OPC	14	AF

Operation

```
PC ← @SP
SP ← SP + 2
```

This instruction is normally used to return from a procedure entered by a CALL instruction. The contents of the location addressed by the SP are popped into the PC. The next statement executed is the one addressed by the new contents of the PC. The stack pointer is also incremented by two.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *Each PUSH instruction executed within the subroutine should be countered with a POP instruction in order to guarantee the SP is at the correct location when the RET instruction is executed. Otherwise the wrong address is loaded into the PC and the program does not operate as appropriate.*

Example

If SP contains `2000h`, external data memory location `2000h` contains `18h`, and location `2001h` contains `B5h`, the statement:

```
RET
Op Code: AF
```

leaves the value `2002h` in the SP, and the PC contains `18B5h`, the address of the next instruction to be executed.

Rotate Left

Syntax

```
RL dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	90	R
		6	91	IR

Operation

```
C ← dst(7)
dst(0) ← dst(7)
dst(1) ← dst(0)
dst(2) ← dst(1)
dst(3) ← dst(2)
dst(4) ← dst(3)
dst(5) ← dst(4)
dst(6) ← dst(5)
dst(7) ← dst(6)
```

The contents of the destination operand are rotated left by one bit position. The initial value of bit 7 is moved to the bit 0 position and also into the Carry Flag, as shown below.
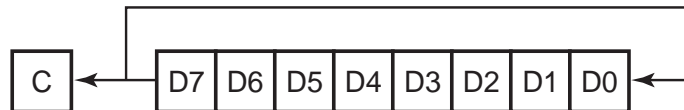
C	←	D7	D6	D5	D4	D3	D2	D1	D0	←

Flag	Description
C	Set if the bit rotated from the most significant bit position was 1 (i.e., bit 7 was 1).
Z	Set if the result is zero; cleared otherwise.
S	Set if the result in bit 7 is set; cleared otherwise.
V	Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D	Unaffected.
H	Unaffected.

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble*

of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the contents of Register C6h are 88h (10001000b), the statement:

```
RL C6h
Op Code: 80 C6
```

leaves the value 11h (00010001b) in Register C6h. The C and V Flags are set, and the S and Z Flags are cleared.

Example 2

If the contents of Register C6h are 88h, and the contents of Register 88h are 44h (01000100b), the statement:

```
RL @C6h
Op Code: 81 C6
```

leaves the value 88h in Register 88h (10001000b). The S and V Flags are set, and the C and Z Flags are cleared.
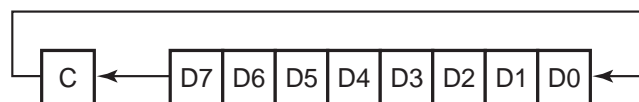
Rotate Left Through Carry

Syntax

```
RLC dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	10	R
		6	11	IR

Operation

```
C← dst(7)
dst(0) ← C
dst(1) ← dst(0)
dst(2) ← dst(1)
dst(3) ← dst(2)
dst(4) ← dst(3)
dst(5) ← dst(4)
dst(6) ← dst(5)
dst(7) ← dst(6)
```

The contents of the destination operand along with the C Flag are rotated left by one bit position. The initial value of bit 7 replaces the C Flag and the initial value of the C Flag replaces bit 0, as shown below.

```
 ┌──────────────────────────────────────┐
 │  ┌───┐    ┌──┬──┬──┬──┬──┬──┬──┬──┐   │
 └──│ C │◄───│D7│D6│D5│D4│D3│D2│D1│D0│◄──┘
    └───┘    └──┴──┴──┴──┴──┴──┴──┴──┘
```

Flag	Description
C	Set if the bit rotated from the most significant bit position was 1 (i.e., bit 7 was 1).
Z	Set if the result is zero; cleared otherwise.
S	Set if the result bit 7 is set; cleared otherwise.
V	Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D	Unaffected
H	Unaffected

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble*

of the operand. For example, if Working Register R12 (CH) is the destination operand, then `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the C Flag is reset and Register C6 contains `8Fh` (`10001111b`), the statement:

```
RLC C6
Op Code: 10 C6
```

leaves Register C6 with the value `1Eh` (`00011110b`). The C and V Flags are set, and S and Z Flags are cleared.

Example 2

If the C Flag is reset, Working Register R4 contains `C6h`, and Register C6 contains `8Fh` (`10001111b`), the statement:

```
RLC @R4
Op Code: 11 E4
```

leaves Register C6 with the value `1Eh` (`00011110b`). The C and V Flags are set, and S and Z Flags are cleared.
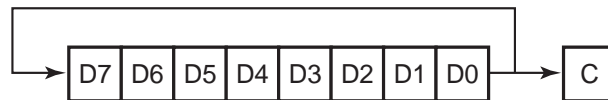
Rotate Right

Syntax

```
RR dst
```

Instruction Format

		OPC	Address Mode
	Cycles	(Hex)	dst

OPC	dst

Cycles	OPC (Hex)	Address Mode dst
6	E0	R
6	E1	IR

Operation

```
C ← dst(0)
dst(0) ← dst(1)
dst(1) ← dst(2)
dst(2) ← dst(3)
dst(3) ← dst(4)
dst(4) ← dst(5)
dst(5) ← dst(6)
dst(6) ← dst(7)
dst(7) ← dst(0)
```

The contents of the destination operand are rotated to the right by one bit position. The initial value of bit 0 is moved to bit 7 and also into the C Flag, as shown below.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | → | C |

Flag Description

C	Set if the bit rotated from the least significant bit position was 1 (that is, bit 0 was 1).
Z	Set if the result is zero; cleared otherwise.
S	Set if the result bit 7 is set; cleared otherwise.
V	Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D	Unaffected
H	Unaffected

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the contents of Working Register R6 are `31h` (00110001B), the statement:

```
RR R6
Op Code: E0 E6
```

leaves the value `98h` (`10011000`) in Working Register R6. The C, V, and S Flags are set, and the Z Flag is cleared.

Example 2

If the contents of Register C6 are `31h` and the contents of Register `31h` are `7Eh` (`01111110b`), the statement:

```
RR @C6
Op Code: E1 C6
```

leaves the value `4Fh` (`00111111`) in Register `31h`. The C, Z, V, and S Flags are cleared.

Rotate Right Through Carry
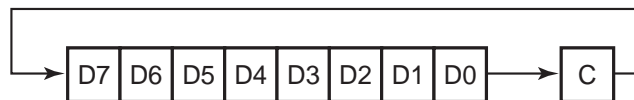
Syntax

```
RRC dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	C0	R
		6	C1	IR

Operation

```
C ← dst(0)
dst(0) ← dst(1)
dst(1) ← dst(2)
dst(2) ← dst(3)
dst(3) ← dst(4)
dst(4) ← dst(5)
dst(5) ← dst(6)
dst(6) ← dst(7)
dst(7) ← C
```

The contents of the destination operand with the C Flag are rotated right by one bit position. The initial value of bit 0 replaces the C Flag and the initial value of the C Flag replaces bit 7, as shown below.

D7	D6	D5	D4	D3	D2	D1	D0	→	C

Flag Description

C   Set if the bit rotated from the least significant bit position was 1 (that is, bit 0 was 1).
Z   Set if the result is zero; cleared otherwise.
S   Set if the result bit 7 is set; cleared otherwise.
V   Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D   Unaffected
H   Unaffected

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble*

of the operand. For example, if Working Register R12 (CH) is the destination operand, then `ECh` *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the contents of Register `C6h` are `DDh` (`11011101b`) and the C Flag is reset, the statement:

```
RRC C6h
Op Code: C0 C6
```

leaves the value `6Eh` (`01101110b`) in register `C6h`. The C and V Flags are set, and the Z and S Flags are cleared.

Example 2

If the contents of Register `2Ch` are `EDh`, the contents of Register `EDh` is ← (`00000000b`), and the C Flag is reset, the statement:

```
RRC @2Ch
Op Code: C1 2C
```

leaves the value `02h` (`00000010b`) in Register `EDh`. The C, Z, S, and V Flags are reset.

Set Carry Flag

Syntax

```
SRC
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	DF

Operation

$C \leftarrow 1$

The C Flag is set to 1, regardless of its previous value.

Flag	Description
C	Set to 1
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example

If the C Flag is currently reset, the statement:

```
SCF
Op Code: DF
```

sets the Carry Flag to 1.

Set Register Pointer

Syntax

```
SRP src
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	src	6	31	IM

Operation

```
RP ← src
```

The specified value is loaded into the Register Pointer (RP) (Control Register FDh). Bits 7-4 determine the Working Register Group. Bits 3-0 selects the Expanded Register Bank. Addressing of un-implemented Working Register Group, while using Expanded Register Banks, points to Bank 0.

Example 1

SRP TD addresses Working Register Group 7 of Bank 0.

Register Pointer (FDh)	Working Register Group	Actual Registers
Contents (Bin)	(Hex)	(Hex)
1111 0000	F	F0–FF
1110 0000	E	E0–EF
1101 0000	D	D0–DF
1100 0000	C	C0–CF
1011 0000	B	B0–BF
1010 0000	A	A0–AF
1001 0000	9	90–9F
1000 0000	8	80–8F
0111 0000	7	70–7F
0110 0000	6	60–6F
0101 0000	5	50–5F
0100 0000	4	40–4F
0011 0000	3	30–3F
0010 0000	2	20–2F
0001 0000	1	10–1F
0000 0000	0	00–0F

Register Pointer (FDh) Contents (Hex)	Expanded Register Bank (Hex)
xxxx 1111	F
xxxx 1110	E
xxxx 1101	D
xxxx 1100	C
xxxx 1011	B
xxxx 1010	A
xxxx 1001	9
xxxx 1000	8
xxxx 0111	7
xxxx 0110	6
xxxx 0101	5
xxxx 0100	4
xxxx 0011	3
xxxx 0010	2
xxxx 0001	1
xxxx 0000	0

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *When an Expanded Register Bank, other than Bank 0 is selected, access to the Z8® Standard Register File is possible except for the Port Register and general purpose registers* 04h *to* 0Fh.

```
fpr Register Addresses 0h to Fh
```

Example 2

The statement:

```
SRP F0h Op Code: 31 F0
```

sets the Register Pointer to access expanded Register Bank 0 and Working Register Group F in the Z8® Standard Register File. All references to Working Registers now affect this

group of 16 registers. Registers `F0h` to `FFh` can be accessed as Working Registers R0 to R15.

Example 3

The statement:

```
SRP 0Fh
Op Code: 31 0F
```

sets the Register Pointer to access Expanded Register Bank F, Reg ← to Reg `0Fh`, as the current Working Registers. All references to Working Registers now affect this group of 16 registers. These registers are now accessed as Working Registers R0 to R15. Port Registers are now not accessable.

Example 4

Assume the RP currently addresses the Control and Peripheral Working Register Group and the program has just entered an interrupt service routine. The statement:

```
SRP 70h
Op Code: 31 70
```

retains the contents of the Control and Peripheral Registers by setting the RP to `70h` (`01110000b`). Any reference to Working Registers in the interrupt routine points to registers `70h` to `7Fh`.

Shift Right Arithmetic

Syntax

```
SRA dst
```

Instruction Format

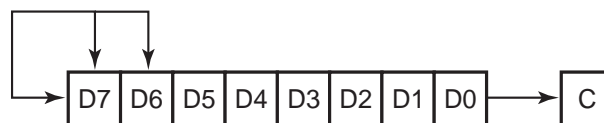|		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	D0	R
		6	D1	IR

Operation

```
C ← dst(0)
dst(0) ← dst(1)
dst(1) ← dst(2)
dst(2) ← dst(3)
dst(3) ← dst(4)
dst(4) ← dst(5)
dst(5) ← dst(6)
dst(6) ← dst(7)
dst(7) ← dst(7)
```

An arithmetic shift right by one bit position is performed on the destination operand. Bit 0 replaces the C Flag. Bit 7 (the Sign bit) is unchanged and its value is shifted into bit 6, as shown below.

Flag Description

Flag	Description
C	Set if the bit rotated from the least significant bit position was 1 (that is, bit 0 was 1).
Z	Set if the result is zero; cleared otherwise.
S	Set if the result bit 7 is set; cleared otherwise.
V	Always reset to 0.
D	Unaffected
H	Unaffected

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, destination Working Register operand is specified by adding* 1110b (Eh) *to the high nibble of*

the operand. For example, if Working Register R12 (CH) *is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If the contents of Working Register R6 are 31h (00110001B), the statement:

```
SRA R6
Op Code: D0 E6
```

leaves the value 98h (00011000) in Working Register R6. The C Flag is set, and the Z, V, and S Flags are cleared.

Example 2

If Register C6 contains the value DFh, and Register DFh contains the value B8h (10111000B), the statement:

```
SRA @C6
Op Code: D1 C6
```

leaves the value DCh (11011100B) in Register DFh. The C, Z, and V Flags are reset, and the S Flag is set.

STOP

Syntax

```
STOP
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	6F

Operation

This instruction turns OFF the internal system clock (SCLK) and external crystal (XTAL) oscillation, and reduces the standby current. STOP mode is terminated by a RESET which causes the processor to restart the application program at address `000Ch`.

Flag	Description
C	Unaffected
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

> **Note:** *In order to enter STOP mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. You must execute a NOP immediately before the execution of the STOP instruction.*

Example

The statements:

```
NOP
STOP
Op Codes: FF 6F
```

place the Z8® CPU into STOP mode.

Subtract

Syntax

```
SUB dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	6	22	r	r
			6	23	r	Ir
OPC	src	dst	10	24	R	R
			10	25	R	IR
OPC	dst	src	10	26	R	IM
			10	27	IR	IM

Operation

```
dst ← dst-src
```

The source operand is subtracted from the destination operand and the result is stored in the destination operand. The contents of the source operand are not affected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flag Description

C   Cleared if there is a carry from the most significant bit of the result; set otherwise, indicating a *borrow*.

Z   Set if the result is 0; cleared otherwise.

V   Set if arithmetic overflow occurred (if the operands were of opposite sign and the sign of the result is the same as the sign of the source); reset otherwise.

S   Set if the result is negative; cleared otherwise.

H   Cleared if there is a carry from the most significant bit of the low order four bits of the result; set otherwise indicating a *borrow*.

D   Always set to 1.

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register* R12 (CH) *is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R3 contains `16h`, and Working Register R11 contains `20h`, the statement:

```
SUB R3, R11
Op Code: 22 3B
```

leaves the value F6h in Working Register R3. The C, S, and D Flags are set, and the Z, V, and H Flags are cleared.

Example 2

If Working Register R15 contains `16h`, Working Register R10 contains `20h`, and Register `20h` contains `11h`, the statement:

```
SUB R16, @R10
Op Code: 23 FA
```

leaves the value `05h` in Working Register R15. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 3

If Register `34h` contains `2Eh`, and Register `12h` contains `1Bh`, the statement:

```
SUB 34h, 12h
Op Code: 24 12 34
```

leaves the value `13h` in Register `34h`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 4

If Register `4Bh` contains `82h`, Working Register R3 contains `10h`, and Register `10h` contains `01h`, the statement:

```
SUB 4Bh, @R3
Op Code: 25 E3 4B
```

leaves the value 81h in Register `4Bh`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 5

If Register `6Ch` contains `2Ah`, the statement:

```
SUB 6Ch, #03h
Op Code: 26 6C 03
```

leaves the value `27h` in Register `6Ch`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 6

If Register `D4h` contains `5Fh`, Register `5Fh` contains `4Ch`, the statement:

```
SUB @D4h, #02h
Op Code: 17 D4 02
```

leaves the value `4Ah` in Register `5Fh`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Subtract With Carry

Syntax

```
SBC dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	6	32	r	r
			6	33	r	Ir
OPC	src	dst	10	34	R	R
			10	35	R	IR
OPC	dst	src	10	36	R	IM
			10	37	IR	IM

Operation

```
dst ← dst-src-C
```

The source operand, along with the setting of the C Flag, is subtracted from the destination operand and the result is stored in the destination operand. The contents of the source operand are not affected. Subtraction is performed by adding the two's complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry (borrow) from the subtraction of low order operands to be subtracted from the subtraction of high order operands.

Flag Description

C   Cleared if there is a carry from the most significant bit of the result; set otherwise, indicating a *borrow*.

Z   Set if the result is 0; cleared otherwise.

V   Set if arithmetic overflow occurred (if the operands were of opposite sign and the sign of the result is the same as the sign of the source); reset otherwise.

S   Set if the result is negative; cleared otherwise.

H   Cleared if there is a carry from the most significant bit of the low order four bits of the result; set otherwise indicating a *borrow*.

D   Always set to 1.

> **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* `1110b` *(*`Eh`*) to the high nibble of the operand. For example, if Working Register* `R12` `(CH)` *is the destination operand, then* `ECh` *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R3 contains `16h`, the C Flag is set to 1, and Working Register R11 contains `20h`, the statement:

```
SBC R3, R11
Op Code: 32 3B
```

leaves the value `F5h` in Working Register R3. The C, S, and D Flags are set, and the Z, V, and H Flags are all cleared.

Example 2

If Working Register R15 contains `16h`, the C Flag is not set, Working Register R10 contains `20h`, and Register `20h` contains `11h`, the statement:

```
SBC R16, @R10
Op Code: 33 FA
```

leaves the value `05h` in Working Register R15. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 3

If Register `34h` contains `2Eh`, the C Flag is set, and Register `12h` contains `1Bh`, the statement:

```
SBC 34h, 12h
Op Code: 34 12 34
```

leaves the value `13h` in Register `34h`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 4

If Register `4Bh` contains `82h`, the C Flag is set, Working Register R3 contains `10h`, and Register `10h` contains `01h`, the statement:

```
SBC 4Bh, @R3
Op Code: 35 E3 4B
```

leaves the value `80h` in Register `4Bh`. The D Flag is set, and the C, Z, S, V, and H flags are cleared.

Example 5

If Register `6Ch` contains `2Ah`, and the C Flag is not set, the statement:

```
SBC 6Ch, #03h
Op Code: 36 6C 03
```

leaves the value `27h` in Register `6Ch`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example 6

If Register `D4h` contains `5Fh`, Register `5Fh` contains 4Ch, and the C Flag is set, the statement:

```
SBC @D4h, #02h
Op Code: 37 D4 02
```

leaves the value `4Ah` in Register `5Fh`. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Swap Nibbles

Syntax

```
SWAP dst
```

Instruction Format

		Cycles	OPC (Hex)	Address Mode dst
OPC	dst	6	F0	R
		6	F1	IR

Operation

dst(7-4) ↔ dst(3-0)

The contents of the lower four bits and upper four bits of the destination operand are swapped.

Flag	Description
C	Unaffected
Z	Set if the result is zero; cleared otherwise.
S	Set if the result bit 7 is set; cleared otherwise.
V	Undefined
D	Unaffected
H	Unaffected

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register* R12 (CH) *is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	dst

Example 1

If Register BCh contains B3h (10110011B), the statement:

```
SWAP B3h
Op Code: F0 B3
```

leaves the value 3Bh (00111011B) in Register BCh. The Z and S Flags are cleared.

Example 2

If Working Register R5 contains `BCh` and Register `BCh` contains `B3h` (10110011B), the statement:

```
SWAP @R5h
Op Code: F1 E5
```

leaves the value `3Bh` (00111011B) in Register `BCh`. The Z and S Flags are cleared.

Test Complement Under Mask

Syntax

```
TCM dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	6	62	r	r
			6	63	r	Ir
OPC	src	dst	10	64	R	R
			10	65	R	IR
OPC	dst	src	10	66	R	IM
			10	67	IR	IM

Operation

```
(NOT dst) AND src
```

This instruction tests selected bits in the destination operand for a logical 1 value. The bits to be tested are specified by setting a 1 bit in the corresponding bit position in the source operand (the mask). The TCM instruction complements the destination operand, and then ANDs it with the source mask (operand). The Zero (Z) Flag can then be checked to determine the result. If the Z Flag is set, then the tested bits were 1. When the TCM operation is complete, the destination and source operands still contain their original values.

Flag	Description
Z	Set if the result is zero; cleared otherwise.
S	Set if the result bit 7 is set; cleared otherwise.
V	Always reset to 0.
D	Unaffected
H	Unaffected

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R3 contains 45h (01000101b) and Working Register R7 contains the value 01h (00000001b) (bit 0 is being tested if it is 1), the statement:

```
TCM R3, R7
Op Code: 62 37
```

sets the Z Flag indicating bit 0 in the destination operand is 1. The V and S Flags are cleared.

Example 2

If Working Register R14 contains the value F3h (11110011b), Working Register R5 contains CBh, and Register CBh contains 88h (10001000b) (bit 7 and bit 3 are being tested if they are 1), the statement:

```
TCM R14, @R5
Op Code: 63 E5
```

resets the Z Flag, because bit 3 in the destination operand is not a 1. The V and S Flags are also cleared.

Example 3

If Register D4h contains the value 04h (000001000b), and Working Register R0 contains the value 80h (10000000b) (bit 7 is being tested if it is 1), the statement:

```
TCM D4h, R0
Op Code: 64 E0 D4
```

resets the Z Flag, because bit 7 in the destination operand is not a 1. The S Flag is set, and the V Flag is cleared.

Example 4

If Register DFh contains the value FFh (11111111b), Register 07h contains the value 1Fh, and Register 1Fh contains the value BDh (10111101b) (bit 7, bit 5, bit 4, bit 3, bit 2, and bit 0 are being tested if they are 1), the statement:

```
TCM DFh, @07h
Op Code: 65 07 DF
```

sets the Z Flag indicating the tested bits in the destination operand are 1. The S and V Flags are cleared.

Example 5

If Working Register R13 contains the value F2h (11110010b), the statement:

```
TCM R13, #02h
Op Code: 66 ED, 02
```

tests bit 1 of the destination operand for 1. The Z Flag is set indicating bit 1 in the destination operand was 1. The S and V Flags are cleared.

Example 6

If Register `5Dh` contains `A0h`, and Register `A0h` contains `0Fh` (`00001111b`), the statement:

```
TCM @5D, #10h
Op Code: 67 5D 10
```

tests bit 4 of the Register `A0h` for 1. The Z Flag is reset indicating bit 1 in the destination operand was not 1. The S and V Flags are cleared.

Test Under Mask

Syntax

```
TM dst, src
```

Instruction Format

			Cycles	OPC (Hex)	Address Mode dst	src
OPC	dst	src	6	72	r	r
			6	73	r	lr
OPC	src	dst	10	74	R	R
			10	75	R	IR
OPC	dst	src	10	76	R	IM
			10	77	IR	IM

Operation

```
dst AND src
```

This instruction tests selected bits in the destination operand for a 0 logical value. The bits to be tested are specified by setting a 1 bit in the corresponding bit position in the source operand (the mask). The TM instruction ANDs the destination operand with the source operand (the mask). The Zero (Z) Flag can then be checked to determine the result. If the Z Flag is set, then the tested bits were 0. When the TM operation is complete, the destination and source operands still contain their original values.

Flag	Description
Z	Set if the result is zero; cleared otherwise.
S	Set if the result bit 7 is set; cleared otherwise.
V	Always reset to 0.
D	Unaffected
H	Unaffected

▶ **Note:** *Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding* 1110b *(*Eh*) to the high nibble of the operand. For example, if Working Register* R12 (CH) *is the destination operand, then* ECh *is used as the destination operand in the Op Code.*

E	src	or	E	dst

Example 1

If Working Register R3 contains `45h` (`01000101b`) and Working Register R7 contains the value `02h` (`00000010b`) (bit 1 is being tested if it is 0), the statement:

```
TM R3, R7
Op Code: 72 37
```

sets the Z Flag indicating bit 1 in the destination operand is 0. The V and S Flags are cleared.

Example 2

Working Register R14 contains the value `F3h` (`11110011b`), Working Register R5 contains `CBh`, and Register `CBh` contains `88h` (`10001000b`) (bit 7 a bit 3 are being tested if they are 0), the statement:

```
TM R14, @R5
Op Code: 73 E5
```

resets the Z Flag, because bit 7 in the destination operand is not a 0. The S Flag is set, and the V Flag is cleared.

Example 3

If Register `D4h` contains the value `08h` (`00001000b`), and Working Register R0 contains the value `04h` (`00000100b`) (bit 2 is being tested if it is 0), the statement:

```
TM D4h, R0
Op Code: 74 E0 D4
```

sets the Z Flag, because bit 2 in the destination operand is a 0. The S and V Flags is cleared.

Example 4

If Register `DFh` contains the value ← (`00000000b`), Register `07h` contains the value `1Fh`, and Register `1Fh` contains the value `BDh` (`10111101b`) (bit 7, bit 5, bit 4, bit 3, bit 2, and bit 0 are being tested if they are 0), the statement:

```
TM DFh, @07h
Op Code: 75 07 DF
```

sets the Z Flag indicating the tested bits in the destination operand are 0. The S is set, and the V Flag is cleared.

Example 5

If Working Register R13 contains the value `F1h` (`11110001b`), the statement:

```
TM R13, #02h
Op Code: 76 ED, 02
```

tests bit 1 of the destination operand for 0. The Z Flag is set indicating bit 1 in the destination operand was 0. The S and V Flags are cleared.

Example 6

If Register `5Dh` contains `A0h`, and Register `A0h` contains `0Fh` (`00001111b`), the statement:

```
TM @5D, #10h
Op Code: 77 5D 10
```

tests bit 4 of the Register `A0h` for 0. The Z Flag is set indicating bit 4 in the destination operand was 0. The S and V Flags are cleared.

Watchdog Timer

Syntax

```
WDT
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	5F

Operation

The WDT is a retriggerable one shot timer that resets Z8® CPU if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction. Each subsequent execution of the WDT instruction refreshes the timer and prevents the WDT from timing out.

Flag	Description
Z	Undefined
S	Undefined
V	Undefined
D	Unaffected
H	Unaffected

> **Note:** *The WDT instruction should not be used following any instruction in which the condition of the Flags is important.*

Example 1

If the WDT is enabled, the statement:

```
WDT
Op Code: .BYTE  5Fh
```

refreshes the Watchdog Timer.

Example 2

The first execution of the statement:

```
WDT
Op Code: .BYTE  5Fh
```

enables the Watchdog Timer.

Watchdog Timer Enable During Halt Mode

Syntax

```
WDh
```

Instruction Format

	Cycles	OPC (Hex)
OPC	6	4F

Operation

When this instruction is executed it enables the WDT during HALT mode. If this instruction is not executed the WDT stops when entering HALT mode. This instruction does not clear the counter, it just makes it possible to have the WDT function running during HALT mode. A WDh instruction executed without executing WDT (5Fh) has no effect.

Flag	Description
Z	Undefined
S	Undefined
V	Undefined
D	Unaffected
H	Unaffected

➤ **Note:** *The WDh instruction should not be used following any instruction in which the condition of the Flags is important.*

Example

If the WDT is enabled, the statement:

```
WDh
Op Code: .BYTE 4Fh
```

enables the WDT in HALT mode.

➤ **Note:** *This instruction format is valid only for the Z86C04, Z86C08, Z86E04, Z86E07, and Z86E08 MCUs.*

Index

A

acknowledge 107
Address Strobe 136
Addressing 132
ART 115

B

Bit Rate 116
Bit-Rate 116
Bus Operation 134
Bus Timing 137
by 16 112

C

Condition 109
Counter 110
Counter/Timer 110

D

Data Memory 134
Data Strobe 136
Descriptions 131
Divide 112
Divide-by-16 112

E

Extended Bus Timing 137
External 97
External Addressing Configuration 132
External Interface 131
External Stacks 133

F

Framing 119

Framing Errors 119

G

general-purpose 115
general-purpose register 115

H

HALT 110
HALT and STOP 110
Halt Mode Operation 110

I

Initialization 99
Input 115
Instruction Timing 138
Interface 131
interrupt acknowledge time 107
Interrupt Cycle 107
Interrupt Generation 104
Interrupt Initialization 99
Interrupt Latency 107
Interrupt Mask Register 101
Interrupt Mask Register Initialization 101
Interrupt Priority 100
Interrupt Priority Register Initialization 100
Interrupt Request 98
Interrupt Request Register Initialization 102
Interrupt Request Register Logic and Timing 98
Interrupt Sources 96
    External Interrupt Sources 97
    Internal Interrupt Sources 98
Interrupts 95
    Disable Interrupt 95
    Enable Interrupt 95
    Interrupt Request Register 95
Introduction 131
IRQ 98

V

Vectored 105
Vectored Interrupt Cycle Timing 107
Vectored Processing 105

W

Worst Case 107

Z

Z8 140
Z8 Reset Conditions 140

Customer Support

For answers to technical questions about the product, documentation, or any other issues with Zilog's offerings, please visit Zilog's Knowledge Base at http://www.zilog.com/kb.

For any comments, detail technical questions, or reporting problems, please visit Zilog's Technical Support at http://support.zilog.com.